A comparison of the finite element and boundary element methods for solving partial differential equations associated with engineering problems

Thesis

UNRESTRICTED

A COMPARISON OF THE FINITE ELEMENT AND BOUNDARY ELEMENT METHODS FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS ASSOCIATED WITH ENGINEERING PROBLEMS

by

Ann Mary Pleasants, M.Sc.

A Thesis submitted to the Faculty of Mathematics with the Open University in the discipline of Computer Science in fulfilment of the requirements for the degree of Master of Philosophy

20th December, 1985

Date of submission : December 1985

Date of award : 11 March 1986

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to thank my external supervisor, Dr. J. F. Baty, for introducing me to these studies and suggesting the topic of research. I also wish to thank him for his support and guidance through the whole period of study.

I wish to thank my internal supervisor, Dr. M. Bromilow, for rescuing me at the 'eleventh hour' and his help with the preparation of the thesis for his suggested improvement in Chapter 3.

STATEMENT OF ORIGINALITY

This thesis is entirely my own work and to the best of my knowledge is fully original except as acknowledged in the text. It has not been submitted in whole or part for a degree, diploma or other award in any university or similar institution.

ABSTRACT

Two numerical methods, the finite element method and the boundary element method, have been compared by studying the elastic torsion problem for various shaped cross sections including ones where there are boundary singularities. A series of numerical experiments was performed illustrating the effects of grid refinement on convergence for each method and a comparison of the amount of work involved in using each method made.

CHAPTER 1

INTRODUCTION

Two different numerical methods both useful for solving the boundary value problems that arise in engineering and physics have been studied - the finite element method and the boundary element method. The computer is essential to both and consequently so are programs that are reliable and accurate. Programs for the study of boundary value problems have been implemented. The results obtained for similar examples have been compared. As both methods were studied by their use the programming has been a major part of the work, particularly in the case of the finite element method where it was necessary to develop routines that could automatically generate a variety of element meshes over the shapes being studied.

Chapter 2 defines concepts used throughout the work. Chapter 3 discusses the solution of Poisson's equation over a rectangle by the finite element method. After doing an example with standard shape functions associated with the rectangular and triangular elements, an investigation was carried out into the possibility of developing a new family of shape functions that approximate the standard ones in the least square sense. These elements would have the advantage of being lower order and it was hoped they would prove more efficient in use. Although they did reduce computer run time in the first cases investigated, ill-conditioned system matrices were generated in others. The

improvement in run time looked better at the time the work was done than it does in retrospect. Another experiment was made into assembling the stiffness matrix by a modified collocation method where again run times were reduced. This preoccupation with computer run time seems dated, but the cost of using computers was then much higher and the cost of program production was overlooked. The situation has now changed. As neither of these techniques results in a method that is any easier to implement or uses less computer store, but both reduce accuracy for a minor improvement in efficiency, the investigation was not continued.

The next chapter, Chapter 4, describes the solution of Laplace's equation over a rectangle by the boundary element technique. The computer results are also produced where the boundary is a circle demonstrating the extra ease with which this method deals with curved boundaries.

The L-shape is the next region investigated using these methods introducing a singularity into the solution at the reentrant corner. The results were poor near this corner in both methods. Grid refinement was used to improve it - in particular increasing the number of elements in the neighbourhood of the singularity. The theoretical justification of this approach in the finite element case has since been found [3]. The same kind of technique produced an improvement in the results for boundary element method.

As an example of a practical problem the well-known torsion of a structural section was studied. Two different models, the

stress function and warping function formulations, are used. The finite element method seemed the natural method for solving the stress function and the boundary element method for finding warping function because the results could be produced with a minimum of extra programming in each case.

Finally, in Chapter 7, the two methods are reviewed and a comparison made. As such a limited range of problems was tackled no general conclusions can be drawn about the two methods. For solving elliptic boundary value problems, particularly those with awkward boundary shapes, the boundary element method is a valuable tool to be used alongside the well-established finite element method.

CHAPTER 2

BASIC THEORY

2.1 Boundary Value Problems [13]

A general second order partial differential equation of the form

$$Lu \equiv a \frac{\partial^2 u}{\partial x^2} + 2b \frac{\partial^2 u}{\partial x \partial y} + c \frac{\partial^2 u}{\partial y^2} + d \frac{\partial u}{\partial x} + e \frac{\partial u}{\partial y} + su = f \qquad (2.1)$$

where the coefficients may be functions of x and y, is said to be elliptic when the condition

$$b^2 - ac < 0$$

is satisfied. Poisson's equation,

$$\Delta u \equiv \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \ , \qquad (2.2)$$

is a well-known example of an elliptic equation and, if $f \equiv 0$, it is known as Laplace's equation.

In practice, a solution is sought satisfying the partial differential equation in a region together with some condition on the boundary. A boundary value problem defined by a partial differential equation is said to be well posed when there exists a unique solution depending continuously on the boundary data. With an elliptic partial differential equation defined in a closed

region R and a single boundary condition at each point of the boundary B this restricts the condition to being one of three forms.

Types of boundary conditions for elliptic problems:

(i) $$u = g_1 \tag{2.3}$$

giving the Dirichlet problem.

(ii) $$\frac{\partial u}{\partial n} = g_2 \tag{2.4}$$

giving the Neumann problem.

(iii) $$\frac{\partial u}{\partial n} + hu = g_3 \tag{2.5}$$

giving the Robin problem.

In all cases it is necessary to assume $a > 0$ and $s \leq 0$ for the problem to have a unique solution. If $s > 0$ there may be an eigensolution to the homogeneous problem so the solution to the non-homogeneous problem is not unique. In cases (ii) and (iii) additional requirements are needed. In (ii), if both $s = 0$ and $f = 0$ the solution is only specified to within an additive constant and in (iii) $h > 0$ is needed to guarantee uniqueness.

2.2 Weighted Residual Method [8]

These are methods for producing numerical solutions to boundary value problems. Suppose a solution is sought to a partial differential equation, written as

$$Lu = f \ , \tag{2.6}$$

satisfying the boundary condition Mu = 0.

If U is an approximate solution of equation (2.6) which satisfies the boundary conditions and has residual,

$$R = LU - f \ , \tag{2.7}$$

such that

$$\int R \, \psi_i \, dA = 0, \ i = 1,2,\ldots,n \ , \tag{2.8}$$

where ψ_i is a set of linearly independent weighting functions the approximate solution approaches the exact solution as n increases. A solution of (2.8) is known as a weak solution. Different weighting functions give rise to a variety of numerical methods.

2.3 Green's Formulae

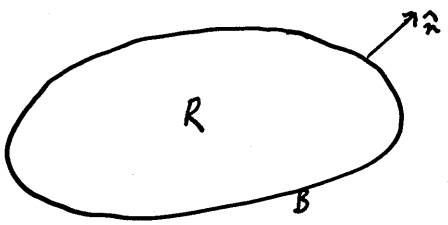


Figure 2.1

In a region R, bounded by piecewise smooth curve B, functions u and v, are related by the formulae

$$\iint_R (u_x v_x + u_y v_y) \, dA + \iint_R v \, \Delta u \, dA = \int_B v \, \frac{\partial u}{\partial n} \, dS \tag{2.9}$$

and

$$\iint_R (u\,\Delta v - v\,\Delta u)\,dA = \iint_B \left(u\,\frac{\partial v}{\partial n} - v\,\frac{\partial u}{\partial n}\right) dS \qquad (2.10)$$

where $\frac{\partial u}{\partial n}$ is the derivative in the direction of the outward normal. Courant[4] assumes that for formula (2.9), u and v are continuous in the closed region R + B, have continuous first derivatives in R and u has a continuous second derivative in R. In the second formula (2.10) he assumes continuity of the first derivatives of u and v in R + B and continuity of the second derivatives in R. There is another useful formula developed from these often called Green's boundary formula[5],

$$cu(P) + \iint v\,\Delta u\,dA = -\int_B \left(u\,\frac{\partial u}{\partial n} - v\,\frac{\partial u}{\partial n}\right) dS \qquad (2.11)$$

where

$$c = \begin{cases} 2\pi & \text{for P inside B} \\ \pi & \text{for P on B} \\ 0 & \text{for P outside B} \end{cases}$$

applying when the boundary R is smooth.

2.4 Analytical Solutions

The analytical solution of a boundary value problem is a function satisfying the partial differential equation at every point of the region R and the boundary condition on boundary B of R. Only a limited number of equations have been solved but there is a solution for the Poisson equation[6],

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -2 \ ,$$

over a rectangle R - $a \leq x \leq a$ and $-b \leq y \leq b$ and

with u = 0 on the boundary of a rectangle x = ± a, y = ± b, found, using Fourier series methods, as

$$u = b^2-y^2 - \frac{32b^2}{\pi^3} \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^3} \operatorname{sech} \frac{(2n+1)\pi a}{2b} \cosh \frac{(2n+1)\pi x}{2b} \cos \frac{(2n+1)\pi y}{2b} \tag{2.12}$$

This series can be used to check values of u found by other methods.

2.5 Numerical Integration over Rectangular Meshes

Throughout the computing whenever numerical integration is performed over straight lines and rectangles, Gaussian quadrature is used. The one-dimensional formula is given by

$$\int_{-1}^{+1} f(\xi)\, d\xi \simeq \sum_{i=1}^{n} W_i\, f(\xi_i) \tag{2.13}$$

where W_i are the weighting factors, ξ_i the coordinates at the i-th sampling point and n the total number of integration points given in Table 2.1. Two dimensional integrals over rectangles are evaluated by using

$$\int_{-1}^{+1} \int_{-1}^{+1} f(\xi,g)\, d\xi\, d\eta \simeq \sum_{j=1}^{n} \sum_{j=1}^{n} W_i\, W_j\, f(\xi_i,\xi_j) \tag{2.14}$$

The points ξ_i and weights W_i have been chosen [7] so that when n sampling points are used any polynomial of degree less than 2n is integrated exactly.

Table 2.1 Gaussian Quadrature Constants

Number of Integrating Points	Coordinates of Integrating Points		Weights at Integrating Points
1	1	0	2
2	1	$1/\sqrt{3}$	1
	2	$-1/\sqrt{3}$	1
3	1	0	8/9
	2	$\sqrt{0.6}$	5/9
	3	$-\sqrt{0.6}$	5/9

CHAPTER 3

FINITE ELEMENT METHOD

3.1 Introduction

There were design problems in the aircraft industry caused when jet engines, with their higher speeds, were introduced and the beginnings of the finite element method arise out of the efforts of structural analysts to solve them. The early work was done entirely by engineers - the subdivision of structures into simple pieces and the connecting equations were based on physical reasoning. As interest in examining the theoretical basis for convergence developed, mathematicians became involved. The process was recognised as an instance of the Rayleigh-Ritz principle and the theory was established on variational principles. Results on convergence, error estimation as well as extensions of the basic method followed with the theory. (A more detailed history can be found in [13]). The role of the computer was vital, not only as the method is ideally suited to a computer implementation but also large scale problems could not be solved otherwise.

As can be seen from the history, the success of the method has come as the result of contributions from 4 main directions:

1. Derivation of the theory.
2. Modelling of actual problems into the finite element framework.
3. Implementation of the theory in computer programs.
4. Improvement of the numerical and computing techniques used.

To obtain this overall view, a simple problem, Poisson's equation over a rectangle, has been studied.

3.2 Solution of Poisson's Equation over a Rectangle

Applying the finite element method to the solution of Poisson's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y) \qquad \ldots\ldots (3.1)$$

over a rectangle, R, satisfying the condition

$$u(x,y) = 0 \qquad \ldots\ldots (3.2)$$

on the boundary, B, of R involves

(i) dividing the rectangle, R, into subregions, called elements, as in Figure 3.1.

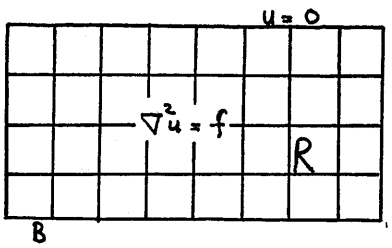


Figure 3.1

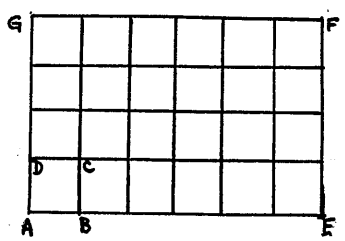
(ii) constructing an approximate solution by choosing its form in each element and through these local representations defining it over the whole region in such a way as to ensure global continuity.

(iii) assembling a set of linear equations to calculate the parameters used in the approximate solution.

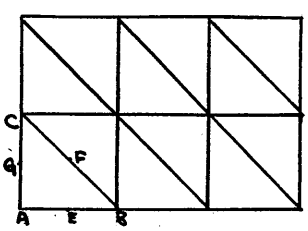
Before solving the problem by the weighted residual method, it is necessary to define some basic concepts.

3.3 Elements[9]

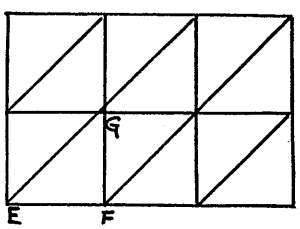
The type of subdivision of a region depends on its shape and the boundary value problem. The elements used here can be either rectangles or triangles, both are simple to use and appropriate for rectilinear regions. In Figure 3.2A a rectangle is subdivided into a mesh of rectangles and in Figures 3.2B and 3.2C into triangular meshes in two different ways.



(a)



(b)



(c)

Figure 3.2

Associated with each element is a set of nodal points (nodes) used when defining the local form of the solution, e.g. the corner points (A,B,C and D) are used as nodes on the rectangles of Figure 3.2A

and on the triangles (EFG) of Figure 3.2C or corner points and midpoints of the sides (A,E,B,F,C,G) of the triangles of Figure 3.2B.

Local Coordinate Systems

When analysing element contributions to the solution it is convenient to work in a local coordinate system related to the element shape.

(i) On rectangular elements, the local coordinate system (ξ,η) is used with the origin O' at the centre of the rectangle and axes are parallel to the sides of the rectangle.

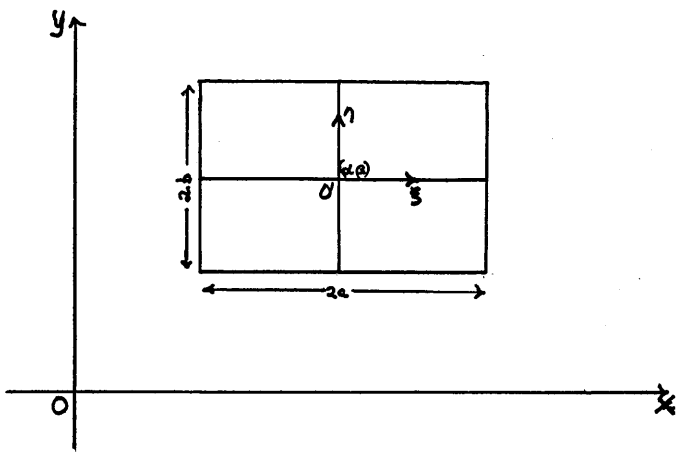


Figure 3.3

The transformation used to convert from global to local coordinates in the rectangular element in Figure 3.3 is

$$\xi = (x-\alpha)/a \text{ and } \eta = (y-\beta)/b \qquad \ldots\ldots (3.3)$$

then $|\xi,\eta| \in [-1,+1]$.

(ii) In triangular elements area coordinates[10], denoted by (L_1, L_2, L_3) are convenient. In the triangle ABC (Figure 3.4) with vertices $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3) the relation between the area coordinates of a point P (L_1, L_2, L_3) and its Cartesian coordinates (x, y) is

$$\begin{aligned} x &= L_1 x_1 + L_2 x_2 + L_3 x_3 \\ y &= L_1 y_1 + L_2 y_2 + L_3 y_3 \\ 1 &= L_1 + L_2 + L_3 \end{aligned} \qquad \ldots\ldots (3.4)$$

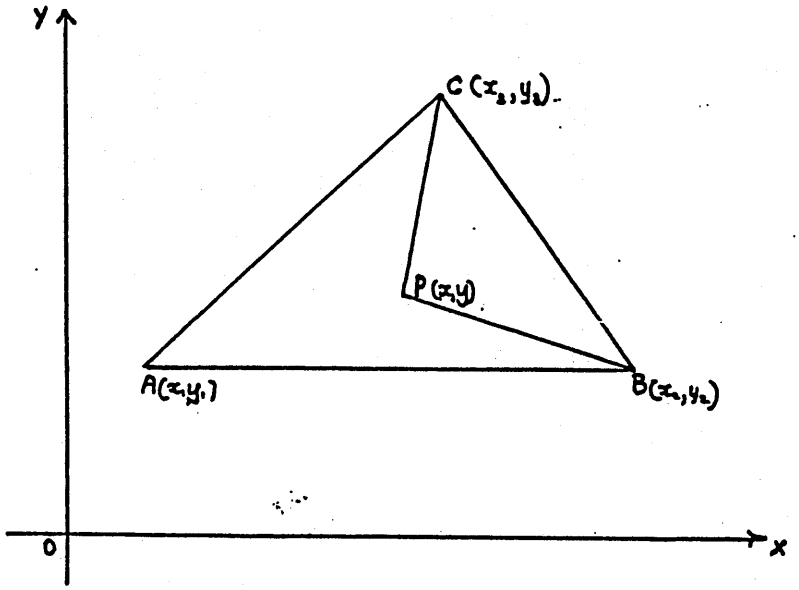


Figure 3.4

The name of the coordinate system (L_1, L_2, L_3) derives from the connection between coordinates and areas as shown in Figure 3.5. That is,

$$L_1 = \frac{\text{Area } \Delta PBC}{\text{Area } \Delta ABC} \qquad \ldots\ldots (3.5)$$

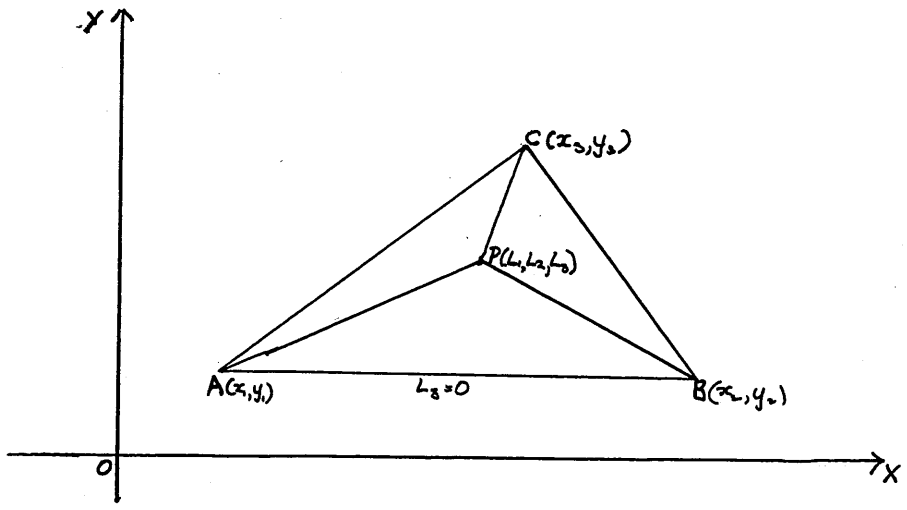


Figure 3.5

The inverse transformation to (3.4) is

$$L_1 = (a_1 + b_1x + c_1y)/2A$$
$$L_2 = (a_2 + b_2x + c_2y)/2A \qquad \ldots\ldots (3.5)$$
$$L_3 = (a_3 + b_3x + c_3y)/2A$$

where $A = \frac{1}{2} \det \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix} = \text{Area } \Delta ABC$

and
$$a_1 = x_2y_3 - x_3y_2$$
$$b_1 = y_2 - y_3 \qquad \ldots\ldots (3.6)$$
$$c_1 = x_3 - x_2$$

and other coefficients are defined cyclically.

Numerical Integration over Triangular Elements[11]

The integrals arising in the analysis involve variables in the limits of integration. A more convenient formulation using area coordinates allows the integral I, an area integral of a function f over a triangle, to be expressed in the form,

$$I = \int_0^1 \int_0^{1-L_1} f(L_1,L_2,L_3)\, dL_1\, dL_2 = \sum_{i=1}^{\lambda} W_i\, f(L_1,L_2,L_3) \qquad \ldots\ldots (3.7)$$

where the coordinates of sampling points and values of weights W_i used in evaluating the summation in (3.7) are given in Table 3.1.

Table 3.1 Numerical Integration Formulae for Triangle

No. of integrating points	i	Triangular coordinates	Weights (W_i)	Order integrated
1	1	1/3,1/3,1/3	1/2	linear
3	1	1/2,1/2,0	1/6	quadratic
	2	0,1/2,1/2	1/6	
	3	1/2,0,1/2	1/6	

3.4 Shape Functions[12]

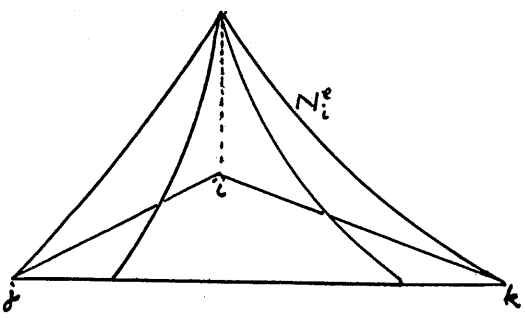


Figure 3.6

On each element a set of nodes is chosen and shape functions are associated with them, allowing the approximate solution, ϕ_e, to be defined within the element. If the shape functions have values at nodes given by

$$N_i^e(x_j,y_j) = \delta_{ij} \qquad \ldots\ldots (3.8)$$

where δ_{ij} is the Kroneker delta.

This allows the approximation within the element to be written as

$$\phi_e(x,y) = \sum_{i=1}^{n} \phi_i N_i^e(x,y) \qquad \ldots\ldots (3.9)$$

where n is the number of nodes in an element

and ϕ_i is the value of ϕ_e at the i-th node.

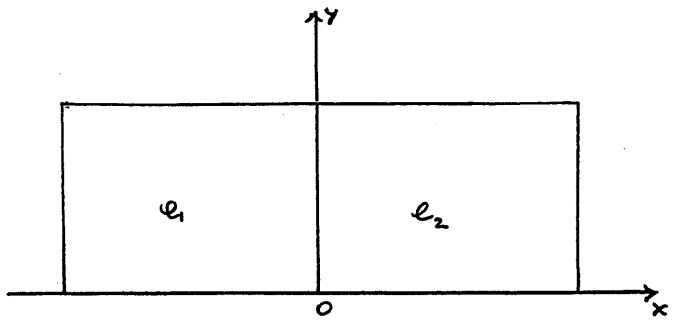


Figure 3.7

If nodes and shape functions are suitably chosen so that along element interfaces the approximate solutions in adjacent elements, i.e. ϕ_{e_1}, ϕ_{e_2}, are equal, then the global solution will

be continuous. If the shape functions, on adjacent elements but associated with a common boundary node, take the same values along the common interface containing that node, then a global basis function can be defined associated with the nodes by

$$N_i(x,y) = \begin{cases} N_i^e(x,y) & \text{if node } i \in e \\ 0 & \text{for all other nodes} \end{cases} \quad \ldots\ldots (3.10)$$

It is now possible to write the global approximate solution, $\phi(x,y)$ as

$$\phi(x,y) = \sum_{j=1}^{N} \phi_j N_j(x,y) \quad \ldots\ldots (3.11)$$

where N is the number of nodes in the region. This approximate solution is used in the weighted residual formulation to find parameters, ϕ_j.

Shape Functions on Rectangular Elements

Bilinear Functions

If the corners are used as nodes in a rectangular element e, where $-1 \leq \xi \leq 1$ and $-1 \leq \eta \leq 1$, the family of functions illustrated in Figure 3.8 and given by

$$N_i^e(\xi,\eta) = 1/4(1 + \xi_o)(1 + \eta_o) \quad \ldots\ldots (3.12)$$

where $\xi_o = \xi\xi_i$, $\eta_o = \eta\eta_i$

and $\xi_i, \eta_i = \pm 1$, $i = 1,\ldots,4$

has the property (3.8). Along each side, the functions, N_i^e, are

linear thus ensuring interelement continuity.

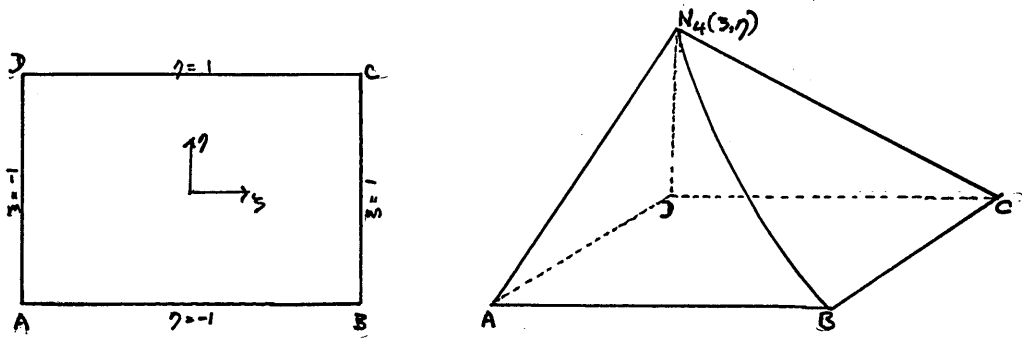


Figure 3.8

The associated global function $N_i(x,y)$ is non-zero on each element containing node i, i.e. on 4 adjacent rectangles except at the boundary (see Figure 3.9).

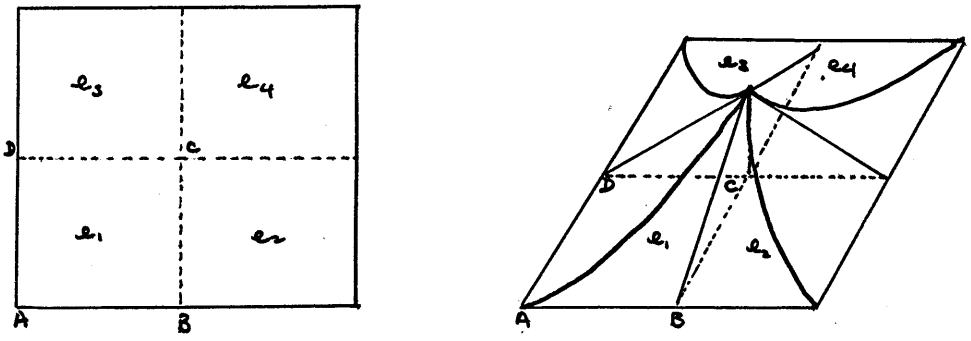


Figure 3.9

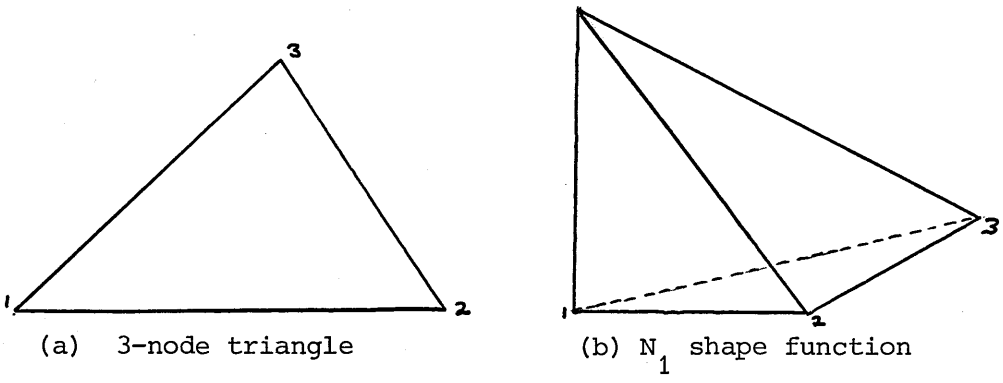
Shape Functions on Triangular Elements

Linear

If the nodes are taken at the vertices of the triangle, the shape functions in element e are given in area coordinates by

$$N_i^e(L_1,L_2,L_3) = L_i \quad \text{where } i = 1,2,3 \qquad \ldots\ldots (3.13)$$

For the special triangulation of a rectangular region as in Figure 3.10C, N_i is non-zero only in the 6 triangles adjacent to node i.



(a) 3-node triangle (b) N_1 shape function

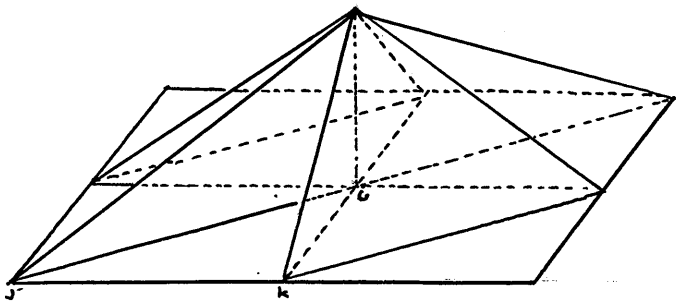


Figure 3.10C

Quadratic

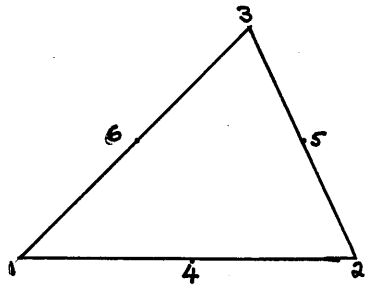
If the nodes are taken at the midpoints of the sides and the vertices of an element e, then a family of quadratic shape functions is defined by the functions,

$$N_i^e = (2L_i - 1)L_i \qquad \text{where } i = 1,2,3 \qquad \ldots\ldots (3.14),$$

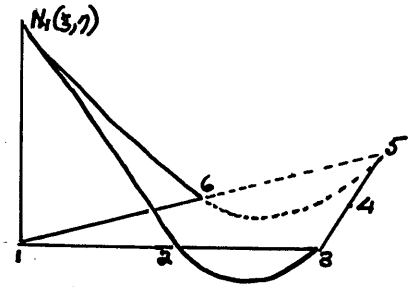
associated with corner nodes A,B,C, and the functions

$$\begin{aligned} N_4^e &= 4\ L_1\ L_2 \\ N_5^e &= 4\ L_2\ L_3 \\ N_6^e &= 4\ L_3\ L_1 \end{aligned} \qquad \ldots\ldots (3.15)$$

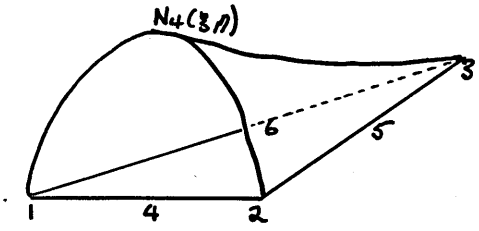
associated with midpoint nodes (4,5 and 6) shown in Figure 3.11a.



(a) 6-node triangle



(b) N_1-corner shape function



(c) N_4-a midpoint node shape function

Figure 3.11

Globally the corner shape functions are non-zero in the adjacent 6 triangular elements while the midpoint functions are non-zero in the 2 adjacent triangles to the node.

3.5 Reduced Shape Functions

The possibility of producing a new set of shape functions by approximating existing families in the least square sense but using the same nodal pattern was investigated. It was hoped that by reducing the order of the shape function polynomial calculation time

could be reduced. The functions were constructed by minimising the integral (in the case of a second order polynomial)

$$I = \iint (N_i^e - (A\xi + B\eta + C))^2 \, d\xi \, d\eta \qquad \ldots\ldots (3.16)$$

where $(A\xi + B\eta + C)$ defines the new shape function, N_i^e, in the element. This process causes the functions to lose the property (3.8) and cease to be linearly independent, but they can be shown to converge to correct values. Bilinear shape functions over rectangles and quadratic ones over triangles were used in the experiment.

Reduced Bilinear Function

After doing the calculation defined by (3.16) the reduced functions used on a rectangle with corner nodes are

$$\bar{N}_i^e = 1/4 \, (1 + \xi\xi_i + \eta\eta_i) \qquad \ldots\ldots (3.17)$$

where $i = 1,2,3,4$

and $\xi_i, \eta_i = \pm 1$

and illustrated in Figure 3.12. It can be seen that the values of the nodes are

$$\bar{N}_i^e = \begin{cases} 3/4 \text{ at node } 1 \\ 1/4 \text{ at node } 2,4 \\ -1/4 \text{ at node } 3 \end{cases} \qquad \ldots\ldots (3.18)$$

Globally this function will have contributions on all the elements adjacent to the element on which it is defined.

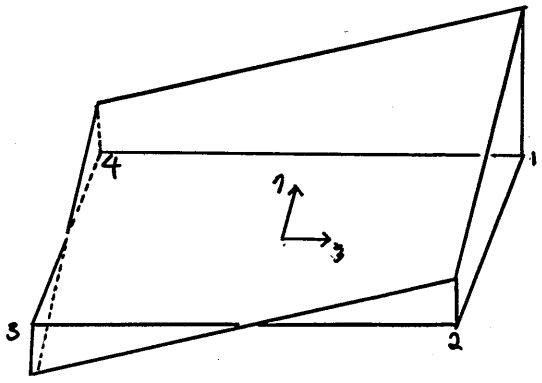


Figure 3.12

Linear Dependence

As the four reduced bilinear functions depend on three parameters they form a linearly dependent set, i.e.

$$1/4(1+\xi+\eta) + 1/4(1-\xi-\eta) = 1/4(1+\xi-\eta) + 1/4(1-\xi+\eta) \quad \ldots\ldots (3.19)$$

$$\bar{N}_1^e + \bar{N}_3^e = \bar{N}_2^e + \bar{N}_4^e \quad \ldots\ldots (3.20)$$

The element matrix calculated using these shape functions is singular.

Convergence

It can be shown that the solution calculated with reduced functions is a reasonable approximation to that calculated with the bilinear shape functions.

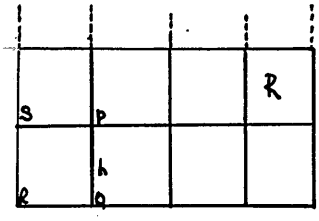


Figure 3.13

Consider the use of reduced bilinear shape functions on a square, R, subdivided into square elements side h.

If the true solution over the square R is ϕ then the finite element approximation ϕ_e which takes the values of ϕ at the grid points can be written, using bilinear shape functions N_i^e, as

$$\phi_e = \sum_{i=1}^{4} N_i^e \phi_i \qquad \ldots\ldots (3.21)$$

expanding this gives

$$\phi_e = 1/4((1+\xi+\eta+\xi\eta)\phi_1+(1-\xi+\eta-\xi\eta)\phi_2+(1+\xi-\eta-\xi\eta)\phi_3+(1-\xi-\eta+\xi\eta)\phi_4) \qquad \ldots\ldots (3.22)$$

Rewriting using reduced shape functions gives

$$\phi_e = \sum_{i=1}^{4} \bar{N}_i^e \phi_i + \frac{\xi\eta}{4} (\phi_1-\phi_2+\phi_3-\phi_4) \ . \qquad \ldots\ldots (3.23)$$

If the approximate solution ϕ_e' is calculated by

$$\phi_e' = \sum_{i=1}^{4} \bar{N}_i^e \phi_i \qquad \ldots\ldots (3.24)$$

the value at P (node 1) is

$$\phi_e'(P) = 3/4\ \phi_1 + 1/4\ (\phi_2+\phi_3) - 1/4\ \phi_4 \qquad \ldots\ldots (3.25)$$

where ϕ_i is the value of the true solution at node i.

Providing the element is sufficiently small, values at the other nodes can be found from it using a Taylor expansion, e.g. the value at S is given by

$$\phi_4 = \phi(S) = \phi_1 - \frac{h\partial\phi_1}{\partial x} + \frac{h^2}{2}\frac{\partial^2\phi_1}{\partial x^2} + O(h^2) \qquad \ldots\ldots (3.26)$$

$$\phi_e'(P) = 3/4\ \phi_1 + 1/4(2\phi_1 - \frac{h\partial\phi_1}{\partial x} + \frac{h^2}{2}\frac{\partial^2\phi_1}{\partial x^2} - \frac{h\partial\phi_1}{\partial y} + \frac{h^2}{2}\frac{\partial^2\phi_1}{\partial y^2})$$

$$- 1/4(\phi_1 - \frac{h\partial\phi_1}{\partial x} - \frac{h\partial\phi_1}{\partial y} + \frac{h^2}{2}(\frac{\partial^2\phi_1}{\partial x^2} + \frac{2\partial^2\phi_1}{\partial x\partial y} + \frac{\partial\phi_1}{\partial y^2})) + O(h^3)$$

$$\ldots\ldots (3.27)$$

$$= \phi_1 - \frac{h^2}{4}\frac{\partial^2\phi_1}{\partial x\partial y} + O(h^3) \qquad \ldots\ldots (3.28)$$

Thus $\phi_e'(P)$ approaches ϕ_1 as h tends to zero.

A similar calculation can be performed at the other nodes, Q, R and S while at the centre of the element ($\xi = 0$, $\eta = 0$) the solution with reduced shape functions equals the exact solution.

Reduced (Triangular) Quadratic Functions

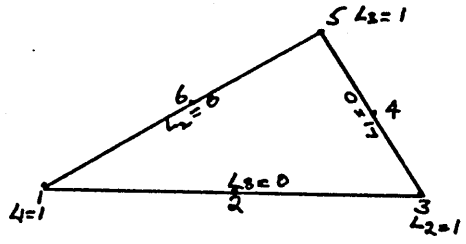


Fig. 3.14

The least square approximation to the quadratic family defined by equations (3.14) and (3.15) is found by minimising the integrals,

$$I = \iint_e (L_j(2L_j-1)-(AL_1+BL_2+CL_3+D))^2 \, de \qquad (j = 1,2,3) \qquad \ldots\ldots (3.29)$$

and

$$J = \iint_e (4L_jL_k-(AL_1+BL_2+CL_3))^2 \, de \qquad (j,k = 1,2,3) \qquad \ldots\ldots (3.30)$$

remembering that $L_1 + L_2 + L_3 = 1$. To calculate the resulting integrals which are expressed in area coordinates the formula (3.31) is used[10].

$$\iint_\Delta L_1^a L_2^b L_3^c \, de = \frac{a!\ b!\ c!}{(a+b+c+2)!} \ 2e \qquad \ldots\ldots (3.31)$$

The calculations yield the functions

$$\bar{N}_i^e = \frac{3L_{2i-1}^{-1}}{5} \qquad \text{when } i = 1,3,5 \qquad \ldots\ldots (3.32)$$

and

$$\bar{N}_2^e = \frac{3-4L_3}{5}$$

$$N_4^e = \frac{3-4L_1}{5} \qquad \ldots\ldots (3.33)$$

$$N_6^e = \frac{3-4L_2}{5}$$

These functions lose the property (3.8), e.g. the values taken by the function N_1^e at the nodes are

$$N_1^e = \begin{cases} 2/5 \text{ at node 1} \\ -1/5 \text{ at nodes 3 and node 4} \\ 1/10 \text{ at node 2} \\ -1/5 \text{ at node 5} \\ 1/10 \text{ at node 6} \end{cases} \qquad \ldots\ldots (3.34)$$

Globally these functions will have contributions on all elements on which they are defined. They also are a linearly dependent set.

Convergence

It can be shown that the solution calculated with the functions in (3.32) and (3.33) is a reasonable approximation to the one calculated with the quadratic (triangular) elements.

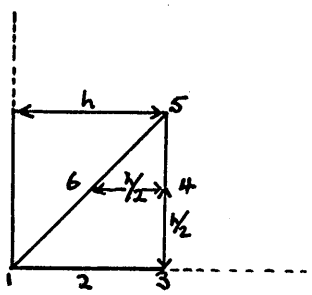


Figure 3.15

A square domain is subdivided into right angled triangles, as in Figure 3.15, and the true solution, ϕ_e, over an element, e, which takes the values of ϕ at the grid points can be written $\phi_e = \sum_{i=1}^{6} N_i^e \phi_i$. If the approximate solution calculated using reduced

shape functions $\bar{N}_i^e$ is $\phi_e' = \sum_{i=1}^{6} \bar{N}_i^e \phi_i$ then the value at P (node 1) is

$$\phi_e'(P) = \frac{2}{5}\phi_1 - \frac{1}{5}(\phi_4 + \phi_3 + \phi_5) + \frac{3}{5}(\phi_2 + \phi_6) \qquad \ldots\ldots (3.35)$$

$\phi_e'(P)$ can be shown to approach the true solution ϕ_1 by using Taylor expansions as for reduced linear functions:

$$\begin{aligned}\phi_e'(P) = \frac{2}{5}\phi_1 &- \frac{1}{5}\left(\phi_1 + h\frac{\partial\phi_1}{\partial x} + \frac{h^2}{2}\frac{\partial^2\phi_1}{\partial x^2} + O(h^3)\right)\\
&- \frac{1}{5}\left(\phi_1 + h\left(\frac{\partial\phi_1}{\partial y} + \frac{\partial\phi}{\partial x}\right) + \frac{h^2}{2}\left(\frac{\partial^2\phi_1}{\partial x^2} + \frac{2\partial^2\phi_1}{\partial x\partial y} + \frac{\partial^2\phi_1}{\partial y^2}\right) + O(h^3)\right)\\
&- \frac{1}{5}\left(\phi_1 + h\frac{\partial\phi_1}{\partial x} + \frac{h}{2}\frac{\partial\phi_1}{\partial y} + \frac{h^2}{8}\left(4\frac{\partial^2\phi_1}{\partial x^2} + \frac{4\partial^2\phi_1}{\partial x\partial y} + \frac{\partial^2\phi_1}{\partial y^2}\right) + O(h^3)\right)\\
&+ \frac{3}{5}\left(\phi_1 + \frac{h}{2}\left(\frac{\partial\phi_1}{\partial y} + \frac{\partial\phi_1}{\partial x}\right) + \frac{h^2}{8}\left(\frac{\partial^2\phi_1}{\partial x^2} + \frac{h^2\partial^2\phi_1}{4\partial x\partial y} + \frac{h^2}{8}\frac{\partial^2\phi_1}{\partial y^2}\right) + O(h^3)\right)\\
&+ \frac{3}{5}\left(\phi_1 + \frac{h}{2}\frac{\partial\phi_1}{\partial x} + \frac{h^2}{8}\frac{\partial^2\phi_1}{\partial x^2} + O(h^3)\right) + \ldots \qquad \ldots\ldots (3.36)\end{aligned}$$

$$= \phi_1 - \frac{h^2}{10}\left(3\frac{\partial^2\phi_1}{\partial x^2} + 3\frac{\partial^2\phi_1}{\partial x\partial y} + \frac{\partial^2\phi}{\partial y^2}\right) + O(h^3) \qquad \ldots\ldots (3.37)$$

Thus $\phi_e(P)$ approaches ϕ_1 as h tends to zero. A similar calculation can be done at each of the other corner nodes Q and R.

At the midpoint node S (node 2) the approximate solution is

$$\phi_e'(S) = \frac{1}{10}(\phi_1 + \phi_3) + \frac{3}{5}\phi_2 + \frac{1}{5}(\phi_4 + \phi_6) - \frac{1}{5}\phi_5 \qquad \ldots\ldots (3.38)$$

Once again

$$\sum_{i=1}^{6} \phi_i = 1$$

and the value $\phi_e'(S)$ approaches ϕ_2 as the side length $h \to 0$. A similar result holds at the other midpoint nodes.

Computing

My supervisor, Dr. J. P. Baty, used these shape functions and obtained a singular system matrix.

3.6 Finite Element Solution Methods

When the approximate solution (3.11) ($\phi = \sum_{i=1}^{N} N_i \phi_i$) is substituted in the weighted residual formula (2.8), this becomes

$$\iint (L \sum_{i=1}^{N} N_i \phi_i - f) \psi_j \, dA = 0 \quad \text{where } j = 1,\ldots,N \qquad \ldots\ldots (3.38)$$

where N is the number of nodes in the rectangle R. If the weighting functions used are the shape functions themselves, i.e. $\psi_j(x,y) = N_j(x,y)$ the Galerkin process results and if they are the Dirac delta functions, i.e. $\psi_j(x,y) = \delta(x-x_j, y-y_j)$, the formula defines the collocation method.

In the case of Poisson's equation and other self adjoint elliptic operators, Green's formula (2.9) can be applied to equation (3.38) giving

$$\iint (L\phi - f)\psi_j \, dA = \iint_R \left(\frac{\partial \phi}{\partial x}\frac{\partial \psi_j}{\partial x} + \frac{\partial \phi}{\partial y}\frac{\partial \psi_j}{\partial y}\right) dA + \oint_B \psi_j \frac{\partial u}{\partial n} \, dS$$

$$- \iint f \psi_j \, dA \qquad \ldots\ldots (3.38a)$$

This widens the choice of functions that can be used as approximate solutions. Working directly with equation (3.38) requires a function $u \in PC^2(R)$ whereas in the weak formulation it is sufficient for $\phi_e \in PC^1(R)$ where $PC^1(R)$ is the set of all piecewise continuous functions on R whose first derivatives are piecewise continuous. The integral over the region R can be replaced by the sum of integrals over the elements into which R has been subdivided giving

$$\sum_e \iint (L\phi - f)\psi_j \, de = \sum_e \iint \left(\frac{\partial \phi}{\partial x} \cdot \frac{\partial \psi_j}{\partial x} + \frac{\partial \phi}{\partial y} \frac{\partial \psi_j}{\partial y}\right) de$$

$$+ \oint_B \psi_j \frac{\partial \phi}{\partial n} \, dS - \sum_e \iint_e f \, \psi_j \, de \quad \text{where } j = 1,\ldots,N \qquad \ldots\ldots (3.39)$$

where the line integral need only be taken around the external boundary. For the reduced shape functions the integrals over internal boundaries no longer cancel.

Galerkin Process

Taking the shape functions, N_j, used in constructing the approximate solution as the weighting functions, (3.39) becomes:

$$\sum_e \iint \left(\frac{\partial \phi}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial \phi}{\partial y} \frac{\partial N_j}{\partial y}\right) de + \oint_B N_j \frac{\partial \phi}{\partial n} \, dS - \sum_e \iint_e f \, N_j \, de = 0$$

$$\text{where } j = 1,\ldots,N \qquad \ldots\ldots (3.40)$$

or

$$\sum_e I_e^j + \oint_B N_j \frac{\partial \phi}{\partial n} \, dS = \sum_e \iint_e f \, N_j \, de \quad \text{where } j = 1,\ldots,N \qquad \ldots\ldots (3.41)$$

where

$$I_e^j = \begin{cases} \iint_e \frac{\partial \phi}{\partial x}\frac{\partial N_j^e}{\partial x} + \frac{\partial \phi}{\partial y}\frac{\partial N_j^e}{\partial y}\, de & \text{if node } j \in e \\ 0 & \text{otherwise} \end{cases} \qquad \ldots\ldots (3.42)$$

Substituting the form (3.11) for ϕ, then

$$I_e^j = \iint_e \sum_i \phi_i \left(\frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial x} + \frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial y}\right) de \text{ where node } i \in e \qquad \ldots\ldots (3.43)$$

and (3.23) becomes a set of N linear equations in the variables ϕ_i

$$\sum_e I_e^j + \sum_e \oint_B N_j \frac{\partial \phi}{\partial n}\, dS = \sum_e \iint_e f\, N_j\, de \qquad j = 1, \ldots, N \qquad \ldots\ldots (3.44)$$

Stiffness Matrix

To rewrite the linear equations (3.44) in matrix form, it is useful to define the quantities

$$k_{ij}^e = \iint_e \frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial x} + \frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial y}\, de\,, \qquad \ldots\ldots (3.45)$$

and $$f_j = \iint_e f\, N_j^e\, de$$

and then (3.24) can be rewritten as

$$\bar{I}_e = k^e\, \bar{\phi}^e \qquad \ldots\ldots (3.46)$$

where $k^e = (k_{ij})$ and node i and node j are in e is a n x n symmetric matrix called the element stiffness matrix, and $\bar{\phi}^e$ is a column vector containing the ϕ-values relating to nodes in element e.

The set of equations (3.43) can be rewritten as

$$K\,\bar{\phi} = F - \oint_B \bar{N}\,\frac{\partial \phi}{\partial n}\,dS \qquad \ldots\ldots (3.47)$$

where K is the system stiffness matrix. It is an N x N symmetric matrix assembled from element matrices, k^e, expanded into N x N matrices by including zeros where a node is not in the element e. F is the column vector $(f_1, f_2, \ldots, f_n)^T$, called the load vector, and $\bar{\phi}$ is $(\phi_1, \phi_2, \ldots, \phi_N)^T$.

Boundary Condition

The integral on the right hand side of equation (3.47) is the only integral involving the boundary.

When the boundary conditions have been prescribed by function values on the boundary, as in (3.2), this is sufficient data to enable all nodal values to be calculated. Other forms of boundary conditions as in 2.4 and 2.5 would require the vector F to be modified by a boundary term.

3.7 Collocation

The conventional collocation, using Dirac delta functions as weighting functions, produces a method that is equivalent to solving the original equation at N points since by substituting an approximate solution ϕ in (2.9) gives, if N is the number of nodes in R,

$$\iint_R (Lu-f)\ \delta(x-x_j,y-y_j)\ dA = 0 \quad \text{where } j = 1,\ldots,N \qquad \ldots\ldots (3.48)$$

and using the property of the δ function that, for any integrable function g

$$\iint g(x,y)\ \delta(x-x_j,y-y_j)\ dA = g(x_j,y_j)\ , \qquad \ldots\ldots (3.49)$$

equation (3.48) becomes

$$L\ u(x_j,y_j) - f(x_j,y_j) = 0, \text{ where } j = 1,\ldots,N \qquad \ldots\ldots (3.50)$$

Modified Collocation

Using the approximate solution in the form (3.11) the shape functions have to have derivatives of the same order at least piecewise as those in L. To use shape functions of lower order it was decided to investigate a modified technique.

In the equation (3.39) resulting from the use of Green's formula on the weighted residual formulation, all integrals are over the element. When using only 1-point Gaussian quadrature which is sufficiently accurate for the integration of linear functions this becomes

$$\sum_e W_1 \left(\frac{\partial\phi}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial\phi}{\partial y}\frac{\partial N_j}{\partial y}\right) = \sum_{j=1}^{N} W_1\ f\ N_j - \oint_B N_j \frac{\partial\phi}{\partial n}\, dS \qquad \ldots\ldots (3.51)$$

where $j = 1,\ldots,N$

and W_1 is the Gaussian weight for 1 point quadrature.

giving a set of N linear equations.

When the shape functions are higher order than linear, the numerical integration is no longer exact. A feasibility study on the technique was carried out and results reported below and in [14].

3.8 Numerical Procedure and Computer Implementation

There are 6 main steps in the numerical analysis of this problem and the associated computer program:

1) input of data
2) generation of mesh
3) assembly of system stiffness matrix
4) application of boundary conditions
5) solution of linear equations
6) output of results and secondary calculations.

Generation of Mesh

The rectangle, R, is subdivided using all rectangular or all triangular elements. A mesh of p x q rectangles is generated by subdividing the length of the rectangle, p times and the width, q times. Triangular elements are generated from a basic rectangular subdivision by dividing the rectangles in half by parallel diagonals. For a six node triangular element the subdivision into 2p x 2q rectangles is made before rearranging into triangles as in Figure 3.16. This generation method is geared to rectangular regions or regions made up of rectangular blocks.

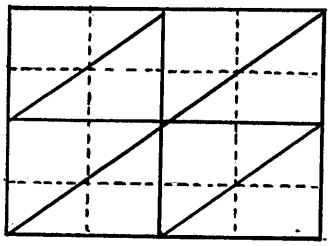


Figure 3.16

Input Data

The data has to specify the following:

(i) geometry of region - either dimensions if automatic generation is being used, or full array of nodal coordinates and nodal/element connection map

(ii) method of solution, i.e. selects element and shape functions to be used, order of numerical integration

(iii) boundary conditions.

Node Connection Map

To keep the bandwidth a minimum, the nodes are numbered in lines parallel to the shortest side of the rectangle as in Figure 3.17. The elements have to be numbered and a map kept of numbers of the nodes associated with each element. These are stored in an array, e.g.

Element	Nodes			
1	1	2	5	6
8	10	11	14	15

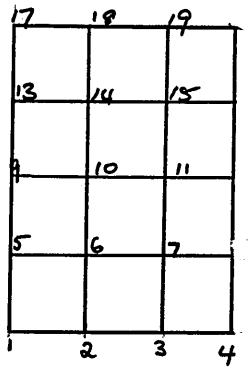


Figure 3.17

Evaluation of a Stiffness Matrix

Element matrix

The evaluation of each contribution k_{ij}^e defined by (3.45) is performed in local coordinates using a transformation from global to local coordinates with Jacobian given by

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{y}{\partial \eta} \end{bmatrix} \qquad \ldots\ldots (3.51)$$

The coefficients k_{ij} involve partial derivatives which are transformed by

$$\begin{bmatrix} \frac{\partial N}{\partial x} \\ \frac{\partial N}{\partial y} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial N}{\partial \xi} \\ \frac{\partial N}{\partial \eta} \end{bmatrix} \qquad \ldots\ldots (3.52)$$

Then

$$k_{ij}^e = \iint_e \left(\frac{\partial N_i^e}{\partial x} \frac{\partial N_j^e}{\partial x} + \frac{\partial N_i^e}{\partial y} \frac{\partial N_j^e}{\partial y} \right) dx\, dy \qquad \ldots\ldots (3.53)$$

$$= \iint_e t(\xi,\eta)\ |J|\ d\xi\, d\eta \qquad \ldots\ldots (3.54)$$

where $t(\xi,\eta)$ is the integrand transformed to (ξ,η) coordinates and $|J|$ is the determinant of the Jacobian, J.

Rectangular elements

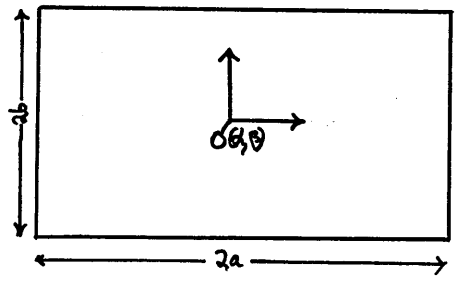


Figure 3.18

Using the local coordinate system defined by

$$\xi = \frac{x-\alpha}{a} \quad , \quad \eta = \frac{y-\beta}{b}$$

the determinant of the transformation is

$$|J| = \begin{vmatrix} a & o \\ o & b \end{vmatrix} = ab$$

Triangular elements

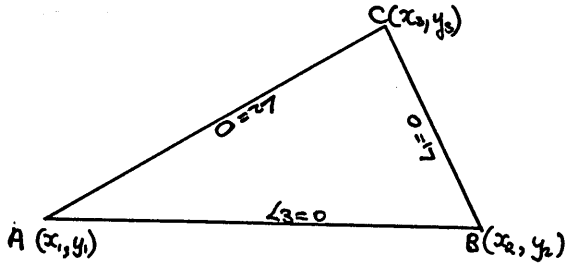


Figure 3.19

As area coordinates are used with triangular elements the Jacobian in the integral (3.54) can be calculated by letting

$$L_1 = \xi$$
$$L_2 = \eta$$
$$L_3 = 1-\xi-\eta$$

and using the relationship given in equations (3.4) rewritten as

$$\begin{pmatrix}1\\x\\y\end{pmatrix} = \begin{pmatrix}1 & 1 & 1\\ x_1 & x_2 & x_3\\ y_1 & y_2 & y_3\end{pmatrix}\begin{pmatrix}\xi\\ \eta\\ 1-\xi-\eta\end{pmatrix}$$

with Jacobian,

$$J = \begin{pmatrix}x_1-x_3 & y_1-y_3\\ x_2-x_3 & y_2-y_3\end{pmatrix}$$

and determinant

$$|J| = (x_1-x_3)(y_2-y_3)-(y_1-y_3)(x_2-x_3)$$

The element stiffness matrix is the matrix k^e defined in equation (3.46) and is for element 2 of example shown in Figure 3.17.

$$\begin{pmatrix}k_{22} & k_{23} & k_{26} & k_{27}\\ k_{32} & k_{33} & k_{36} & k_{37}\\ k_{62} & k_{63} & k_{66} & k_{67}\\ k_{72} & k_{73} & k_{76} & k_{77}\end{pmatrix}$$

Figure 3.20

System Stiffness Matrix

As the element matrices are symmetric only the upper triangular matrix need be calculated. The system matrix is an N x N matrix (where N is the total number of nodes over the region) is assembled by adding the element matrix in such a way that terms in the element matrix correspond to the appropriate nodes and filling with zeros as illustrated in Figure 3.21.

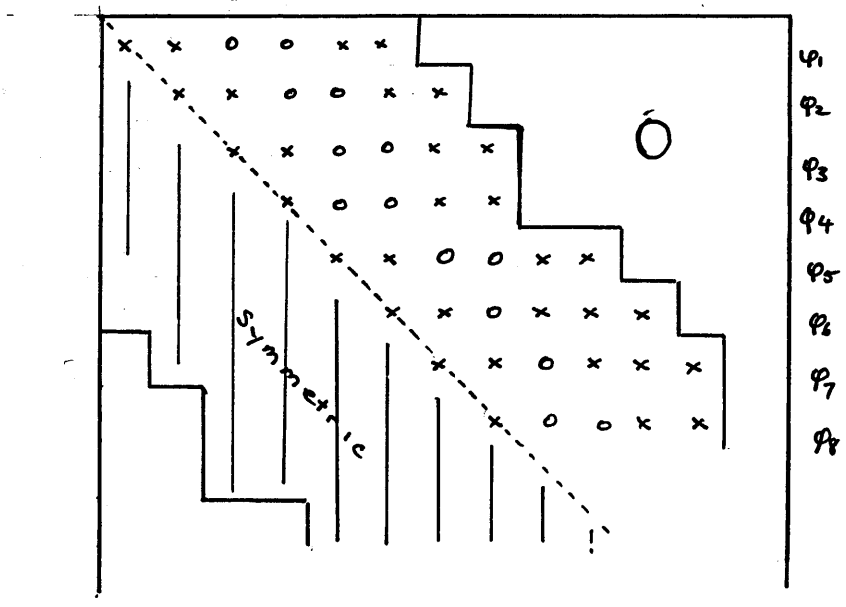


Figure 3.21

The storage of this matrix is related to the solution method. The upper half only is stored in a 1-dimensional array, A, and then only the significant band. It is stored column by column, the first non-zero element in the column being the first stored and an array containing the row number of this first non-zero element is kept.

In the example illustrated in Figure 3.19 where the nth item in the array corresponds to the nth column of the matrix, this array contains

(1,1,2, 3 ,1,1,2,3,5, ...)

As the matrix is stored the position of the diagonal element in each column is recorded in an array DA and an example of Figure 3.19 DA contains

(1,3,5,7,12,18)

The element K_{ij} of the stiffness matrix K is thus stored in position DA(j) + i-j of the array A.

Application of Boundary Conditions[15]

The program can only deal with boundary conditions of the forms u = constant and $\frac{\partial u}{\partial n}$ = 0 on the boundary. To apply the boundary conditions it is desirable, from the programming point of view, not to have to rearrange the stiffness matrix. Instead the diagonal term corresponding to a specified value of u is replaced by unity and the rest of the terms in its column by zero, while the right hand side position is set to the value.

Problem 3.1: To solve $\nabla^2 u = -2$ in the square region shown in Figure 3.22, such that $u = 0$ on the boundary. For each element, with associated shape function, the mesh was refined to investigate convergence

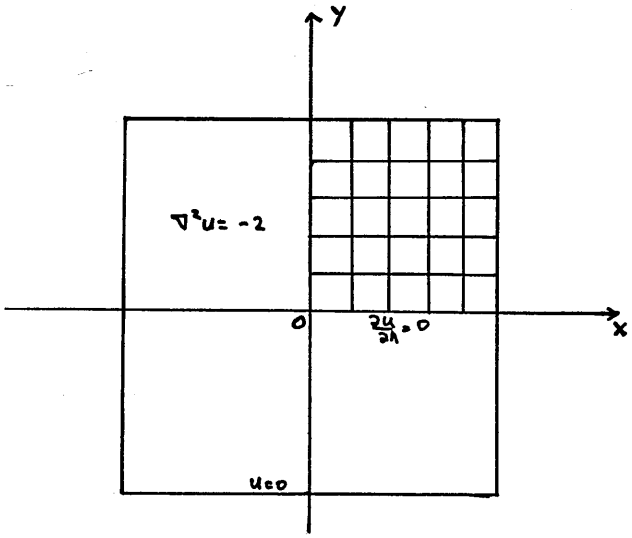


Figure 3.22

Method

The symmetry of the problem in the given region means that it is only necessary to solve over a quarter square by specifying $u = 0$ on the outer boundary and $\frac{\partial u}{\partial n} = 0$ is then implicitly assumed on the inner boundaries. Rectangular elements, as shown in Figure 3.22, with bilinear shape functions and reduced bilinear functions were used.

Rectangular elements 16 x 16

Table 3.4

Method	Shape Function	u(0,0)	Order of numerical integration	Time in etus on 4-50
Galerkin	bilinear	.5898	3	113.80
	reduced bilinear	.5903	3	109.90
	bilinear	.5898	1	45.05
Modified Collocation	bilinear	.5898	1	45.05
	reduced bilinear	.5903	1	45.05
Exact Solution		.5894		

Comment

The figures in Table 3.4 for the Galerkin method highlight the importance of not using a higher order of numerical integration than is necessary. The time saved using the reduced element is much less than that saved by modified collocation - 3.9 etus compared with 88.75 etus with the Galerkin method (3 point integration) and no difference using modified collocation. This latter method is only a saving due to the reduced order of integration.

When the same problem was run using triangular elements with quadratic shape functions and modified collocation, it gave an ill-conditioned system matrix. There were only 32 elements in the mesh so it was due to computing at the limits of the problem. Using the same technique on 1-dimensional problems with cubic shape functions, Dr. J. P. Baty also obtained ill-conditioned matrices. This unreliability in the method caused us to abandon its investigation.

Note: These results (Table 3.4) have been published in a Conference Proceedings by Dr. J. P. Baty and myself [14].

CHAPTER 4

BOUNDARY ELEMENT METHOD

4.1 Introduction

Theoretical solutions to problems in many branches of applied mathematics have involved an integral equation but the formulation of a numerical method based on them had to wait for the computer. The first solutions come in the work of Jaswon[16], Symm[17], followed by Rizzo[18] and Cruse[19]. The method has recently benefitted from publicity, due largely to Brebbia[2] and become known to the engineering community as the boundary element method. It is referred to under other names as well, one of the most common being the boundary integral method. Although there are connections with the finite element method[26], the theoretical approaches are quite different.

Boundary value problems need the following properties[20] if they are to be solved by the method as presented in this chapter: -

(1) Ellipticity of the governing equations.

(2) Existence of a fundamental solution of the partial differential equation.

(3) Reciprocal relation (e.g. Green's theorem) between two solutions.

The basis of the method, first exploited by Jaswon and Symm[16], is that:

(4) The boundary conditions may not be arbitrarily specified and the relation they must satisfy is expressed as a boundary integral equation.

4.2 Solution of Laplace's Equation over a Plane Region

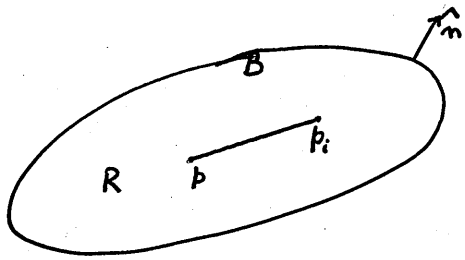


Figure 4.1

The boundary element method is illustrated by solving the boundary value problem defined by Laplace's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \equiv \nabla^2 u = 0 \qquad \ldots\ldots (4.1)$$

over the region R and satisfying the condition;

$$a\, u + b \frac{\partial u}{\partial n} = c \qquad \ldots\ldots (4.2)$$

on the boundary B. It can be derived as a weighted residual method when the fundamental solutions corresponding to the partial differential equation are used as weighting functions. The fundamental solution[25] corresponding to (4.1) is the solution of the equation

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = \delta_i(p,p_i) \qquad \ldots\ldots (4.3)$$

where $\delta_i(p,p_i)$ is Dirac delta function

$\delta_i(p,p_i)$ equals 0 when $p \neq p_i$ and is singular when $p = p_i$

such that $\int_R \delta(p,p_i)\, dA = 1.$

This solution is known to be

$$w(p,p_i) = -\frac{1}{2\pi} \log |\underline{p} - \underline{p}_i| \qquad \ldots\ldots (4.4)$$

where $\underline{p}$ and $\underline{p}_i$ are position vectors of p and p_i.

Starting from the second of Green's formulae (2.10),

$$\iint_R (w \nabla^2 u - u \nabla^2 w)\, dA = \int_B (u \frac{\partial w}{\partial n} - w \frac{\partial u}{\partial n})\, dS \qquad \ldots\ldots (4.5)$$

and substituting for u and w from the equations (4.1) and (4.3) gives

$$-\iint_R u\, \delta_i(p,p_i)\, dA = \int_B (u \frac{\partial w}{\partial n} - w \frac{\partial u}{\partial n})\, dS \qquad \ldots\ldots (4.6)$$

The explicit formula (4.4) is used for w to give

$$-2\pi\, u(p_i) = \int_B \log|\bar{p} - \bar{p}_i| \frac{\partial u}{\partial n} - u \frac{\partial}{\partial n} \log |\bar{p} - \bar{p}_i|\, dS \qquad \ldots\ldots (4.7)$$

which is an integral equation apparently giving the solution to (4.1) at an interior point p_i when p is any point on S. Although this is a boundary integral equation it involves both forms of boundary condition (2.3) and (2.4) on the boundary when both cannot be prescribed arbitrarily for a well posed problem. This equation does not, as yet, provide a means of calculation.

If the point p_i is allowed to approach the boundary at P from the inside where Green's boundary formula (2.11) applies then

$$\pi u(P) = \int_B \log|p-P| \frac{\partial u}{\partial n} dS - \int_B u \frac{\partial}{\partial n} \log |p-P| dS. \quad \ldots\ldots (4.8)$$

This equation may be regarded as defining the relation between the values of u and $\frac{\partial u}{\partial n}$ on the boundary and converted into an effective means of calculating unknown values from prescribed values. When values on the boundary are known (4.7) can be used as a means of finding the values at the interior points of R.

4.3 Numerical Method

To obtain a numerical method of solving (4.1), the boundary of R is discretized as in Figure 4.2 into N subdivisions Γ_j [17], called boundary elements, allowing the integrals on the right hand side of (4.8) to be approximated by a summation. By making some assumption about the behaviour of u over these elements, the integral on each segment can be evaluated thus giving a set of linear equations relating the unknown boundary data and the known.

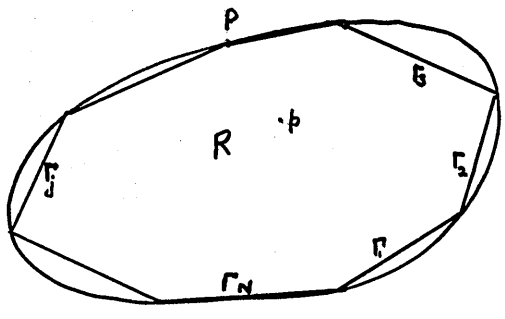


Figure 4.2

Approximating the boundary by N straight line segments Γ_j (j = 1,...,n) equation (4.8) is discretized and an approximate solution, U, to the problem is given by

$$\pi U(P) = \sum_{j=1}^{N} \int_{\Gamma_j} \log |P-p| \frac{\partial U}{\partial n} dS - \int_{\Gamma_j} U \frac{\partial}{\partial n} \log |P-p| dS \quad \ldots\ldots (4.9)$$

To calculate the integrals in equation (4.9) it is necessary to make assumptions about the behaviour of the approximate solution U and $\frac{\partial U}{\partial n}$ over each element on the boundary. Here they are assumed to be either constant or linear.

Constant interpolation[21]

When u and $\frac{\partial u}{\partial n}$ are taken to be constant over each element (constant elements), the integrals on the right hand side of (4.9) can be calculated. If (4.9) is rewritten as

$$\pi U(P) = \sum_{j=1}^{N} U(P_j) I_{1j}(P_j,P) - \sum_{j=1}^{N} \frac{\partial U(P_j)}{\partial n} I_{2j}(P_j,P) \quad \ldots\ldots (4.10)$$

where $I_{1j} = -\int_{\Gamma_j} \frac{\partial}{\partial n} \log |\underset{j}{P}-P| dS$ and $I_{2j} = -\int_{\Gamma_j} \log |\underset{j}{P}-P| dS$ ..... (4.11)

the integrals I_{1j} and I_{2j} can either be calculated numerically or in closed form. When the point P under consideration lies outside an integral Γ_j Gaussian quadrature is used but when it is in the interval the closed form is used to cope with the singularity. Although this should provide sufficient accuracy[2] most implementations of the method seem to use closed form integration throughout for maximum

accuracy. These are obtained by standard integration techniques, relevant formulae being given in [28].

Linear Equations

For a well posed problem equation (4.10) is a system of linear equations which are now written in matrix notation to aid computation, that is

$$H\,\bar{U} = G\,\frac{\partial \bar{U}}{\partial n} \qquad \ldots\ldots (4.12)$$

where $\bar{U}$ is column vector, $\bar{U} = (U_1, U_2, \ldots, U_n)^T$

and $$\frac{\partial \bar{U}}{\partial n} = \left(\frac{\partial U_1}{\partial n}, \frac{\partial U_2}{\partial n}, \ldots, \frac{\partial U_n}{\partial n}\right)^T$$

with

$$H_{ij} = \begin{cases} I_{ij}(P_j, P_i) \text{ when } i \neq j \\ -\pi + I_{1i}(P_i, P_i) \end{cases} \qquad \ldots\ldots (4.13)$$

and $\quad G_{ij} = I_{2j}(P_j, P_i)$ all i,j

At this stage, depending on the form of the boundary data, some values of $\bar{U}$ are known and some of $\frac{\partial \bar{U}}{\partial n}$. It is necessary to rearrange the terms in equation (4.12) so all the unknown values of U and $\frac{\partial U}{\partial n}$ are on the left hand side in the vector $\bar{x}$ and the specified values on the right hand side in $\bar{y}$ giving

$$A\bar{x} = B\bar{y} \qquad \ldots\ldots (4.14)$$

The matrix A obtained by this method is densely packed and not symmetric.

The linear equations can be solved by any standard method - Gaussian elimination was used here.

Linear interpolation[22]

If U and $\frac{\partial U}{\partial n}$ are assumed to be linear on each element having nodes at each end, then they can be expressed in terms of their nodal values by using the interpolation functions,

$$\phi_1 = 1/2\ (1-\xi) \text{ and } \phi_2 = 1/2\ (1+\xi)\ , \qquad \ldots\ldots (4.15)$$

where ξ is the local coordinate along the element as illustrated in Figure 4.3, that is

$$U = U_1\ \phi_1 + U_2\ \phi_2 \qquad \ldots\ldots (4.16)$$

and

$$\frac{\partial U}{\partial n} = \left(\frac{\partial U}{\partial n}\right)_1 \phi_1 + \left(\frac{\partial U}{\partial n}\right)_2 \phi_2\ . \qquad \ldots\ldots (4.17)$$

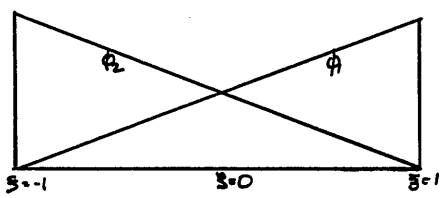


Figure 4.3

The integrals in (4.9) can be written as

$$\int_{\Gamma_j} U \frac{\partial}{\partial n} \log |P-P_i|\ dS = \sum_{k=1}^{2} h_{jk}\ U_k$$

where $h_{jk} = \int_{\Gamma_j} \phi_k \frac{\partial}{\partial n} \log |P-P_i|\ dS$ $\qquad \ldots\ldots (4.18)$

$$\int_{\Gamma_j} \log |P-P_i| \frac{\partial U}{\partial n} dS = \sum_{k=1}^{2} g_{ik} \left(\frac{\partial U}{\partial n}\right)_k$$

where $g_{jk} = \int_{\Gamma_j} \phi_k \log |P-P_i| \, dS$ ..... (4.19)

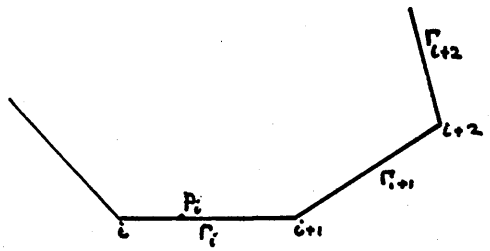


Figure 4.4

Substituting these in equation (4.10) and collecting the terms relating to each node gives

$$U_i + \sum_{j=1}^{n} \bar{H}_{ij} U_j = \Sigma \bar{G}_{ij} \frac{\partial U_j}{\partial n} \quad \text{..... (4.20)}$$

where $\bar{H}_{ij} = h_{i2} + h_{(i+1)1}$ all $i \neq n$

$H_{n1} = h_{n2} + h_{11}$

and G_{ij} are defined similarly.

After obtaining the equations (4.20) they are rearranged for computational purposes as for the constant interpolation case.

This form of linear interpolation corresponds to the program given by Brebbia in [2] and implemented here to study the method. Making the assumption that both U and $\frac{\partial U}{\partial n}$ are linear on an element is inconsistent - an improvement would be to assume $\frac{\partial U}{\partial n}$ is constant.

4.4 Non Smooth Boundaries

When the boundary is not smooth at a point then the equation corresponding to (4.9) relating boundary values becomes

$$\gamma U = \sum_{j=1}^{N} \int_{\Gamma_j} U \frac{\partial w}{\partial n} \, dS - \int_{B} w \frac{\partial U}{\partial n} \, dS \qquad \ldots\ldots (4.21)$$

where the value of γ is the vertex angle. The value is calculated in the program using an idea credited to Cruse in [27] where the particular case of constant potential $U \equiv 1$ on the boundary is considered. Equation (4.21) becomes

$$\gamma U = -\int_{B} U \frac{\partial w}{\partial n} \, dS \qquad \ldots\ldots (4.22)$$

giving for equation (4.18)

$$HU = 0 \qquad \ldots\ldots (4.23)$$

If all components of U are 1 then (4.23) is satisfied when

$$\gamma h_{ii} = - \sum_{\substack{j=1 \\ j \neq 1}}^{n} h_{ij} \quad \text{and } \gamma \text{ can be calculated.}$$

4.5 Poisson's Equation

When partial differential equation in the boundary value problem to be solved is Poisson's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y)$$

the result of applying Green's formula as was done to obtain equation (4.6) has to be modified to become

$$-\iint_R u\delta_i(p,p_i)\ dA + \iint_R f\ w\ dA = \int_S (u\ \frac{\partial w}{\partial n} - w\ \frac{\partial u}{\partial n})\ dS \qquad \ldots\ldots\ (4.24)$$

This area integral, $\iint_R f\ w\ dA$, has to be carried through the work and calculated numerically. In general, this requires the construction of a grid over the region suitable for the numerical integration which has to be done if the computer implementation is to be generally applicable. But in certain cases the problem can be changed to one of solving Laplace's equation by adding a particular solution and making necessary adjustments to the boundary conditions.

4.6 Computer Implementation

The main steps in the numerical procedure are:

1. Input of data.
2. Discretization of the boundary.
3. Handling boundary conditions.
4. Integration over elements and assembly of matrices.
5. Solution of system of linear equations.
6. Calculation of solution at internal points.
7. Output of results.

These steps are reflected in the structure of the computer program implementing the procedure. The main calculation sections are taken from the Fortran subroutines, published by Brebbia in [2], for solving boundary value problems associated with Laplace's equation.

To these have been added routines to simplify the input of data for the boundaries of polygons (a rectangle is given special treatment in view of their simplicity) and circles by performing the discretization of the boundary automatically, and also to solve Poisson's equation, $\nabla^2 u = c$, when c is a constant. The structure of the program and summary is given in Appendix 2.

Input of Data

The data contains information about the type of solution and type of equation being solved but the bulk of the data concerns the mesh generation on the boundary. This can be done by specifying the endpoints of all elements and the boundary value being specified at each node. Although this data is simple in format it is tedious and time consuming to prepare. The program can output the input data for checking purposes. The coordinates of the internal points of the region at which the value of the solution is required have also to be specified as part of the input data.

Boundary Mesh Generation

For a rectangle, the input data specifies the corner coordinates and the number of elements along the boundary edges. It is divided into elements by passing round the boundary in an anti clockwise direction, the nodal positions and corners are recorded.

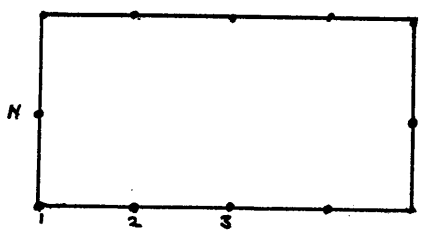


Figure 4.5

In the circular boundary case the input specifies the radius and the number of elements allowing the angle at the centre subtended by each element to be calculated. The elements are a straight line approximation to the boundary and the nodal positions are stored in anticlockwise order.

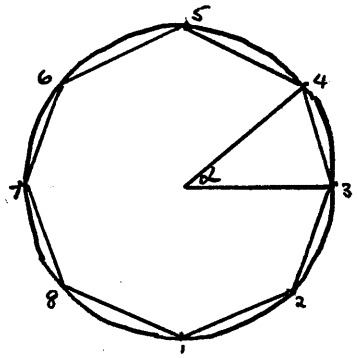


Figure 4.6

For polygons, the straight line boundary specifications are input in anticlockwise order. For each side, the data consists of the number of elements required, corner coordinates and the angle the outward normal makes with the x axis. Nodal positions and corners are recorded. The sides of such are either the geometric sides, e.g. AB and BC in Figure 4.7a, or are sections along which boundary data is unchanged, e.g. AP, PO, OQ and QB allowing for graded meshes to be generated which were used to obtain results on regions with boundary singularities, in Figure 4.7a.

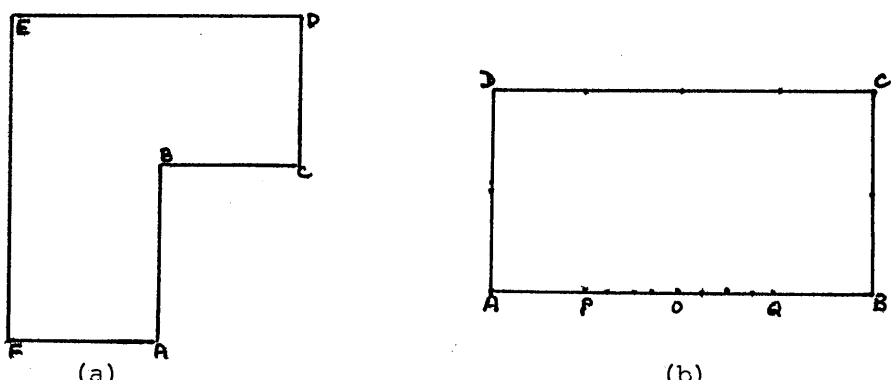


Figure 4.7

Boundary Data

The program needs a boundary value or the value of the normal derivative at each point when a Dirichlet or Robin problem (see 2.1) is being solved. For a Neumann problem, the normal derivative is specified but the value of the function has to be given at least one point or where there is symmetry in the problem at sufficient points to ensure that the symmetry is present in the solution.

Evaluation of Matrices

Two subroutines are used to do the element integration. For the off-diagonal terms of G and H in equation (4.13), the integration is done using a 4 point Gauss formula. Gaussian weights are appropriate to an interval [-1,1] so it is necessary to transform from local coordinates. The integrals in equation (4.14) become

$$I_{2j} = -\sum_{k=1}^{4} \ln |P-P_j|$$

$$I_{1j} = -\sum_{k=1}^{4} \frac{d}{|P_j-P|^2} W_k \, \ell/2 \qquad \ldots\ldots (4.25)$$

where ℓ is length of element

d is perpendicular distance from P_i to element

W_k are Gaussian weights

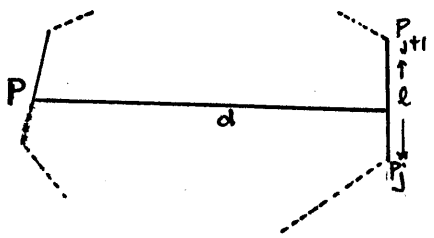


Figure 4.8

For diagonal terms, the formula (4.26) is used

$$G_{ii} = \frac{\ell}{2}\left(\ln\left|\frac{2}{\ell}\right| + 1\right) \quad \text{where } \ell \text{ is length of the element} \qquad \ldots\ldots (4.26)$$

These terms are placed in the matrices G and H as they are calculated. Finally the system is rearranged for solving (as in 4.14), by using the boundary data. The terms corresponding to specified boundary values are swapped between G and H before the solution procedure commences.

Solution of Linear Equations

This is a standard subroutine taken exactly from [2] for solving linear equations $A\bar{x} = B$ by Gaussian elimination with essential row interchanges only (interchanging if the diagonal element is less than 10^{-6} in magnitude). It would have been preferable to use partial pivoting to avoid induced instability. However, no computational problems arose in the solution of the equations.

Evaluation of Internal Points

The equation (4.7) is discretized as above and the same integration routines are used to calculate the values of the required internal points.

Output of Results

At the conclusion of all calculations the values of the solution and normal derivatives are output at all nodes and the

solution value only at those points specified.

Problem 4.1

To solve Poisson's equation $\nabla^2 u = -1$ (in order to compare with results quoted by Jawson and Symm) over the square region illustrated in Figure 4.9 so that the solution u satisfies the condition u = 0 on the boundary

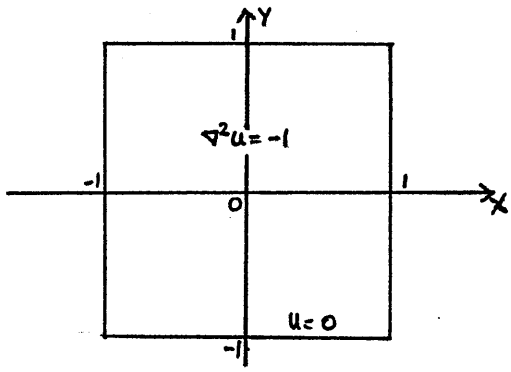


Figure 4.9

A particular solution of (4.4) is

$$p = -1/4\ (x^2 + y^2)$$

so let $\phi = p + u$ ..... (4.27)

then the problem reduces to solving Laplace's equation

$$\nabla^2 \phi = 0$$

with the boundary condition

$$\phi = 1/4\ (x^2 + y^2) \quad \text{..... (4.28)}$$

Results

Table 4.1 compares the computed results for several values of n using linear interpolation over elements with values computed by Jaswon and Symm[28]. Jaswon and Symm's results should be more accurate as they use analytic integration in the evaluation of coefficients in the linear equations. The limits of the program and machine in this problem were reached with a mesh of 80 boundary elements.

Table 4.1 Comparison of results of $\nabla^2\phi = -1$ or $-1 \leq x \leq 1$, $-1 \leq y \leq 1$

Point		Solution on various meshes					
X	Y	N = 32	48	64	80	U^{J+S}_{128}	U_{anal}
0.0	0.0	.2961	.2953	.2950	.2948	.2947	.2947
0.0	0.5	.2308	.2300	.2297	.2295	.2293	.2293
0.5	0.5	.1826	.1818	.1815	.1813	.1811	.1811
.75	.75	.0744	.0735	.0732	.0730	.0728	.0728

In an attempt to reduce the mesh size with the same number of elements and take advantage of the symmetry the above problem was also computed on the quarter square with boundary conditions as shown in Figure 4.10. Only half as many elements were needed to get the same mesh size - a significant improvement in the results.

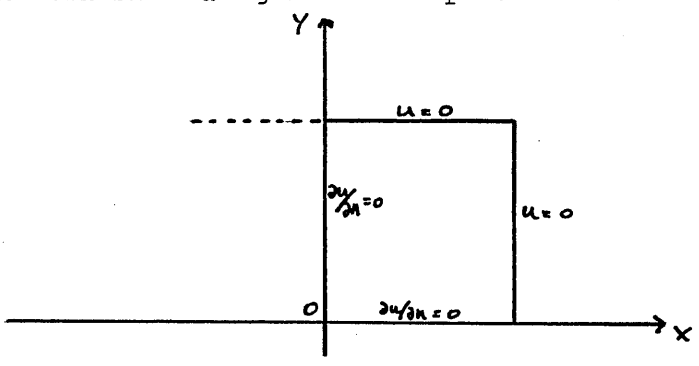


Figure 4.10

Table 4.2 Solution $\nabla^2\phi = -1$ using symmetry shown in Figure 4.10

Point		Solution on various meshes				
x	y	U_{32}	U_{48}	U_{64}	U_{80}	U_{anal}
0.0	0.0	.2950	.2948	.2948	.2948	.2947
0.0	0.5	.2295	.2294	.2294	.2294	.2293
0.5	0.5	.1813	.1812	.1812	.1812	.1811
0.75	0.75	.9731	.0730	.0729	.0729	.0728

Comment

This method does not take full advantage of symmetry but could be done without modification of routines for rectangles. Danson[24] describing BEASY copes with symmetry by reflecting the boundary about the plane of symmetry and continuing the boundary integration around the reflected part of the structure. By this method the time taken to compute the matrices is increased but data preparation reduced. Jaswon and Symm[16,17] take full account of symmetry to reduce equations but give no account of their method.

Problem 4.2

To solve Poisson's equation $\nabla^2\phi = -2$ over the circle, $x^2 + y^2 \leq 1$ with $\phi = 0$ on the boundary.

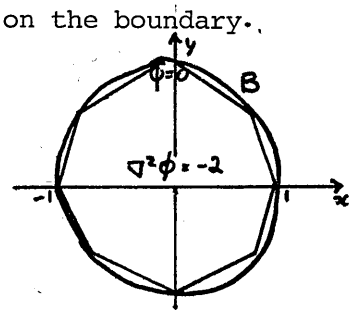


Figure 4.11

The boundary was divided into n arcs of the same length and then approximated by chords. Polar coordinates are convenient for doing this discretization.

Results

n	Interpolation	u(0,0)	u(.5,0)	$\frac{\partial u}{\partial n}$ (B)
8	constant	.5	.375	-1
12	constant	.5	.375	-1
12	linear	.5	.375	-1

The results are exact (to 3 decimal places) but the problem is trivial over the region. The exact solution is $1/2 - 1/2\ (x^2+y^2)$ and the $1/2\ (x^2 + y^2)$ is the particular solution used. This problem was only useful for program testing.

Problem 4.3

To solve Poisson's equation $\nabla^2 u = -2$ where u = 0 on the boundary on the rectangular shapes illustrated in Figure 4.4 (symmetry was not used in this example)

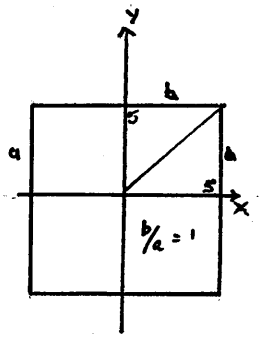


Figure 4.12a

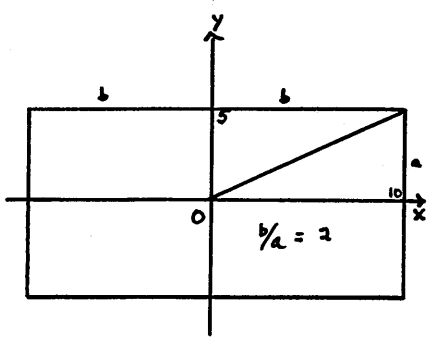


Figure 4.12b

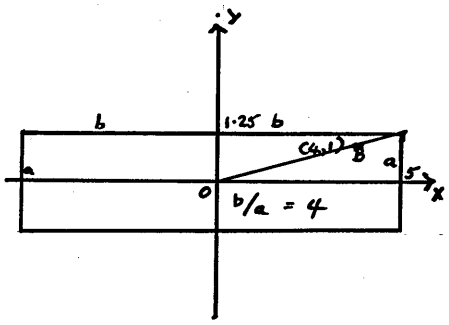


Figure 4.12c

Table 4.3 Comparison of results on rectangles in Figure 4.12

		Solution at point	Calculated	Analytic	Difference
64	1	u(O)	.5912	.5894	.05%
60	2/1	u(O)	22.8171	22.7789	.16%
		u(B)	4.6443	4.6075	.80%
60	4/1	u(O)	1.5982	1.5568	2.66%
		u(B)	.4727	.4200	12.54%

Results

The calculated solution is compared with the analytic solution at the centre of each rectangle and at a point near the boundary. There is an increase in the difference between the two solutions in similar positions as the rectangle becomes narrower and also as the boundary is approached in the same rectangle.

CHAPTER 5

PROBLEMS WITH BOUNDARY SINGULARITIES

5.1 Introduction

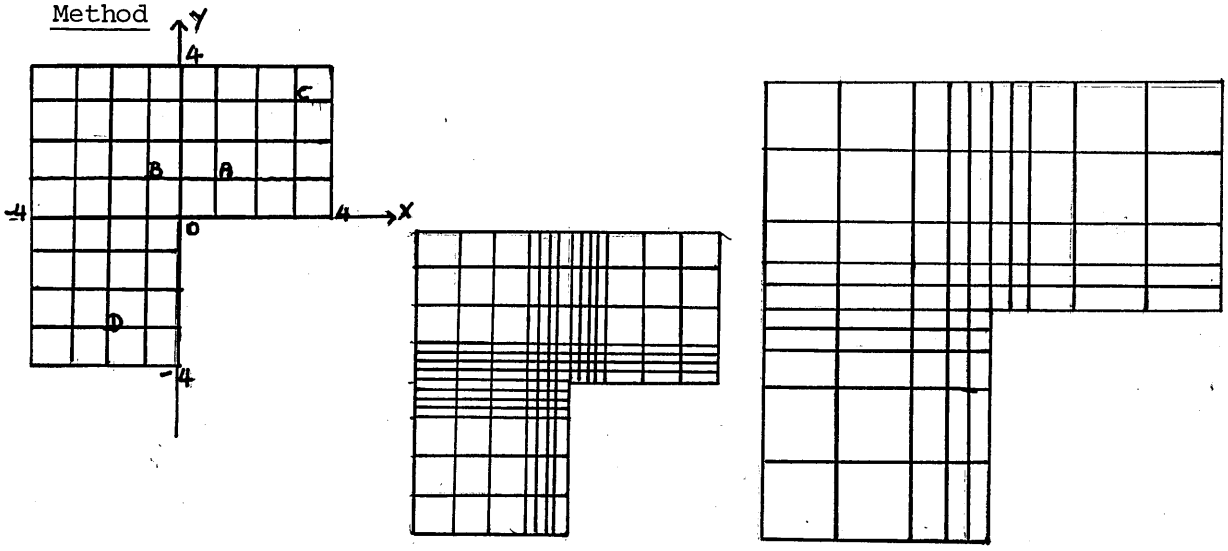
Both the finite element method and the boundary element method suffer from convergence problems with decreasing mesh size near a boundary singularity[37]. The slowing down of the convergence rate is sufficiently serious for the results to be unacceptable under computing conditions used here. Much is known about the nature of singularities that occur in elliptic problems and there are various ways in which the approximate solution can be improved when computing near one. In the finite element method, the shape functions can be modified with appropriate singular functions[36]. While in the boundary element method subtracting out the singular behaviour is effective. Local mesh refinement has been used successfully with the finite element method[3] and it is this approach that has been adopted here and extended to use with boundary elements. The results indicate that it is very effective producing results comparable with those published on two standard problems.

5.2 Finite Element Method

Problem 5.1

Find the solution of $\nabla^2 u = -2$ over L-region, shown in Figure 5.1(a), satisfying $u = 0$ on the boundary.

Method



Uniform Mesh
Figure 5.1(a)

Concentrated Mesh
Figure 5.1(b)

Graded Mesh
Figure 5.1(c)

Results

Table 5.1 Comparison of the effect of mesh type in Problem 5.1

No. of elements	Mesh type	A u(0,1)	B u(-1,1)	C u(3,3)	D u(-2,-3)
148	a	3.7474	4.4293	1.7611	2.2382
192	a	3.7187	4.5562	1.7067	2.1948
147	b	3.7754	4.6168	1.7637	2.2404
108	c	3.7783	4.6102	1.7639	2.2408
300	c	3.7592	4.6005	1.7082	2.1970
Boundary at Solution					
60	c	3.7796	4.6424	1.7423	2.2294

Comment

In Table 5.1 there is clearly the problem of knowing the correct solution so a comparison (Table 5.12) is made with a solution found by the boundary element method using a grid later shown to give

good results. The problem near the origin shows up in the table at the point A where it can be seen that the value computed with a uniform mesh is further from the boundary element value than are the values found at other points. Theoretical work done by Schatz and Wahlbin[3] shows how to construct several kinds of mesh including graded ones that improve convergence near the singularity and indicate what refinement is necessary.

Program Modification

To form complex patterns of elements over the regions of interest by inputting data would have been a time-consuming and error prone process so it was decided to do it by program. As the element generation and node connection routines of the original program were dependent on the geometry of the rectangle it was decided to form complex patterns built up from rectangular units. Some examples of the shapes are shown in Figure 5.2.

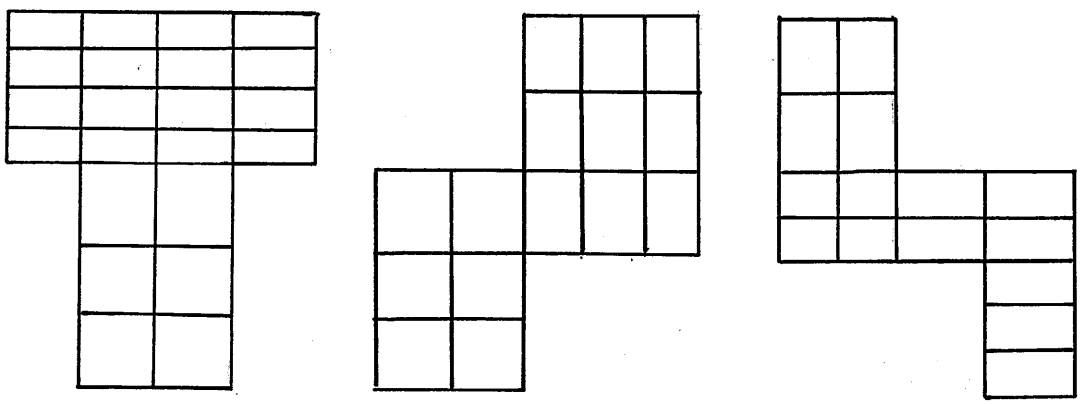


Figure 5.2

The computing falls into three parts: -

(a) Generation of coordinates and nodal patterns over a rectangle.

(b) Formation of nodal pattern over a larger region by building up from these basic units.

(c) Stiffness matrix assembly and solution of the linear equations.

The program has three main routines and data files on disc passing the data between them. Originally, after the testing stage, it had been hoped to make these three steps invisible to the user but this would have required a much larger allocation of disc space. The ways of adding another rectangle to the original shaded rectangle are illustrated in Figure 5.3. It is necessary for elements and nodes to match along the interface and for elements in the basic blocks to be the same size.

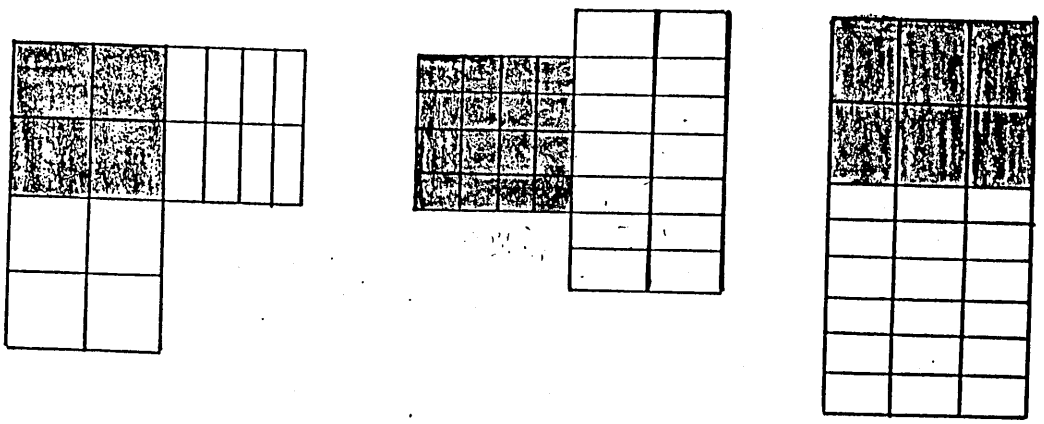


Figure 5.3

Complicated mesh patterns can also be built up and the regions used in generating the mesh for problem 5.1 are shown. The work done in constructing the meshes of Figures 5.4(b) (176 nodes) and (c) (341 nodes) would be reduced if the original generation routine were to allow for elements of variable size.

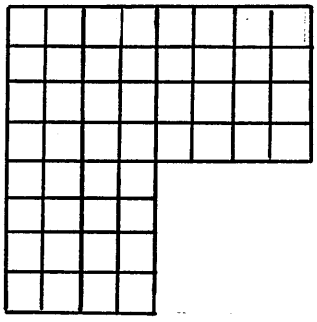


Figure 5.4(a)

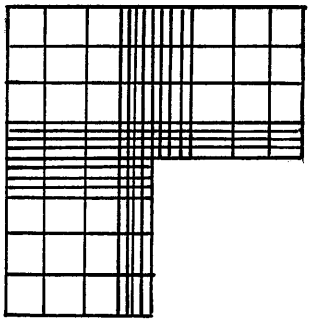


Figure 5.4(b)

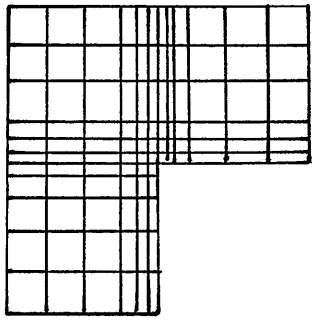


Figure 5.4(c)

5.3 Boundary Element Method

When the boundary element method, as described in Chapter 4, is applied to regions where there are boundary singularities, due either to the geometry of the region or to discontinuities in the boundary conditions, the rate of convergence with decreasing mesh size slows down. Symm[37] uses a modified version of Jaswon's integral equation method which takes the behaviour of the solution near the singularity into account and produces an improvement in both convergence and accuracy. His results over a range of problems are compared with those obtained by Papamichael and Whiteman[38] using a conformal transformation method claimed to be the "most accurate available".

In [37] Symm expands the harmonic function as a power series, in powers of r, about the point of singularity, O, i.e.

$$\phi(P) = f_o(\bar{p}) + \alpha f_1(\bar{p}) + \beta f_2(\bar{p}) + \gamma f_3(\bar{p}) + \ldots \qquad \ldots\ldots (5.1)$$

where P has position vector $\bar{p}$ with polar coordinates (r,θ). The method is illustrated by the following example.

Problem 5.2(a)

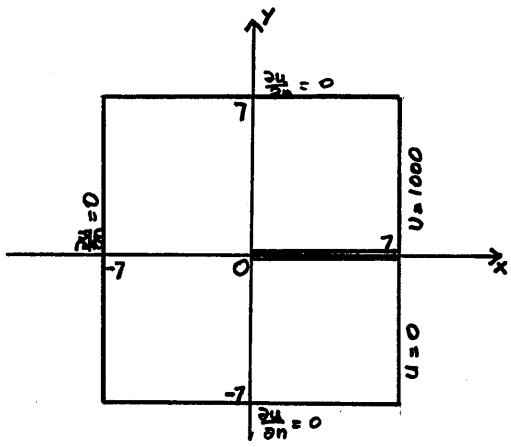


Figure 5.5(a)

The problem is to solve Laplace's equation on a square region with a slit and boundary condition as shown on Figure 5.5(a). This problem was originally studied by Motz[39] using relaxation methods and has become a standard example in singularity studies.

At the centre of the square O there is a sharp re-entrant corner near which Motz[39] takes a solution of the form

$$u(p) = A_o + A_1 r^{1/2} \cos\frac{\theta}{2} + A_2 r \cos\theta + A_3 r^{3/2} \cos\frac{3\theta}{2} + \ldots \qquad \ldots\ldots (5.2)$$

By taking account of symmetry the problem can be reformulated[37] as one where the boundary is smooth but the boundary condition changes

at the point O as shown in Figure 5.5(b). In this case A_o = 500 and $A_2 = A_4 = \ldots = 0$. It is this simpler problem that is considered and to solve it Symm uses the following version of equation 5.1:

$$u(p) = v(p) + \alpha\, r^{1/2} \cos(\theta/2) + \beta\, r^{3/2} \cos(3\theta/2) \qquad \ldots\ldots (5.3)$$

where v(p) satisfies Laplace's equation in R with boundary conditions involving the unknowns α and β.

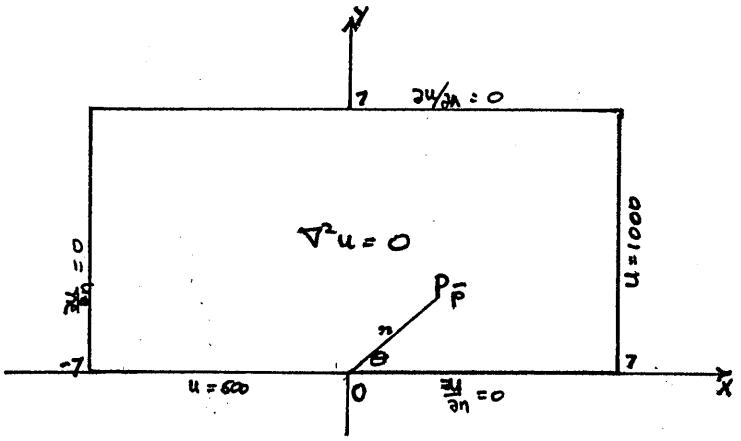


Figure 5.5(b)

The next step in the method is to find an approximation to v(p), α and β by applying the boundary integral method to find v using a mesh with n elements resulting in n equations with n + 2 unknowns. Very close to the origin it is noted that v satisfies the boundary conditions for u giving the two extra equations, i.e. if there are n mesh points

$$\frac{\partial v_1}{\partial n} = 0 \text{ and } V_n = 500 \qquad \ldots\ldots (5.4)$$

Grid Refinement

As an alternative to this method an attempt was made to increase the convergence using a variable mesh, uniform over most of the boundary but denser near the singularity. The nodes, near the singularity O, are either arranged at uniform distances, as in Figure 5.6(a), or graded as in Figure 5.6(b). The number of extra nodes necessary to produce acceptable convergence was investigated experimentally.

Concentrated Mesh

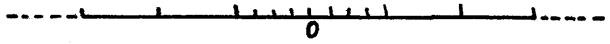


Figure 5.6(a)

Graded Mesh

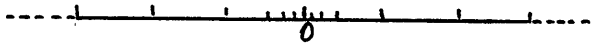
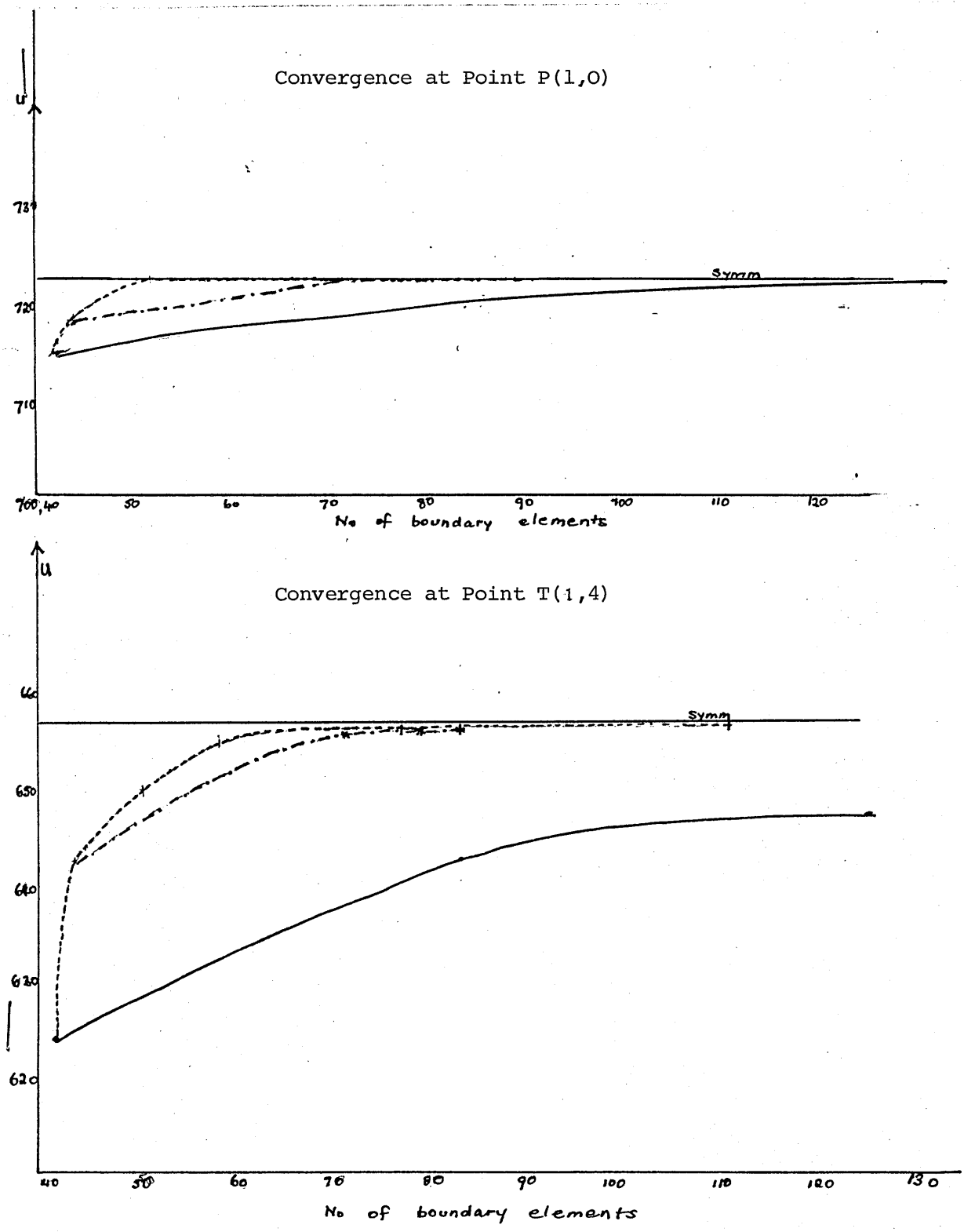


Figure 5.6(b)

The examples worked indicate that a graded mesh produces an improvement with a small number of extra nodes. The effect can be seen on the graph of Figure 5.7 when only one extra point is placed near the singularity (n = 44). With further refinement both types of mesh produce a similar solution.

Figure 5.7 Comparison of Convergence Rates using Different Grids when Solving Problem 5.2b.



Problem 5.2(b)

The problem 5.2 described above was computed using the boundary element method with a variety of meshes. Having nodes at each end of the element made the application of boundary conditions more accurate so linear interpolation was used. The version solved is shown in Figure 5.5(b).

Uniform Mesh

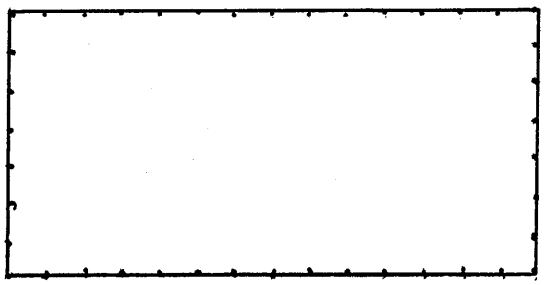


Figure 5.8

A sequence of runs was made dividing the boundary into equal parts and the results are given in Table 5.2. It can be seen that the convergence towards the values obtained by Symm in [36] is slow and particularly poor in the neighbourhood of the origin.

Graded Meshes

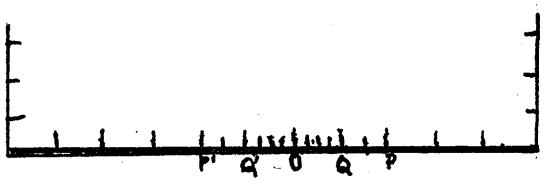


Figure 5.9

Starting with a uniform mesh around the boundary of the rectangle ABCD a graded mesh was constructed by the subdividing of the element on either side of the origin (e.g. OP) into intervals

OQ and QP as shown in Figure 5.9. Each interval was further subdivided equally into elements so elements within any interval were twice as long as those on the interval nearer the origin.

Concentrated Meshes

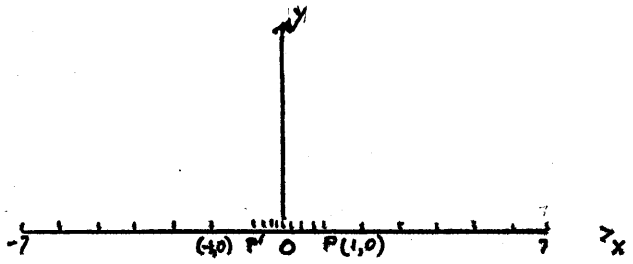


Figure 5.10

For this mesh the boundary was divided into equal elements and the elements on either side of the origin O were further subdivided into smaller elements all the same size.

Results

Uniform Mesh

With the finest grid used (126 nodes) the results in Table 5.2 show a difference from the standard results of about .4% over most of the grid but increasing at the point P(1,0) to 1.3%.

Graded Mesh

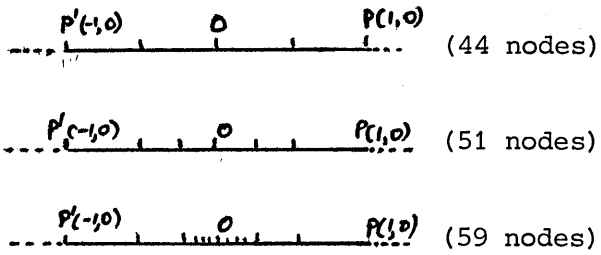
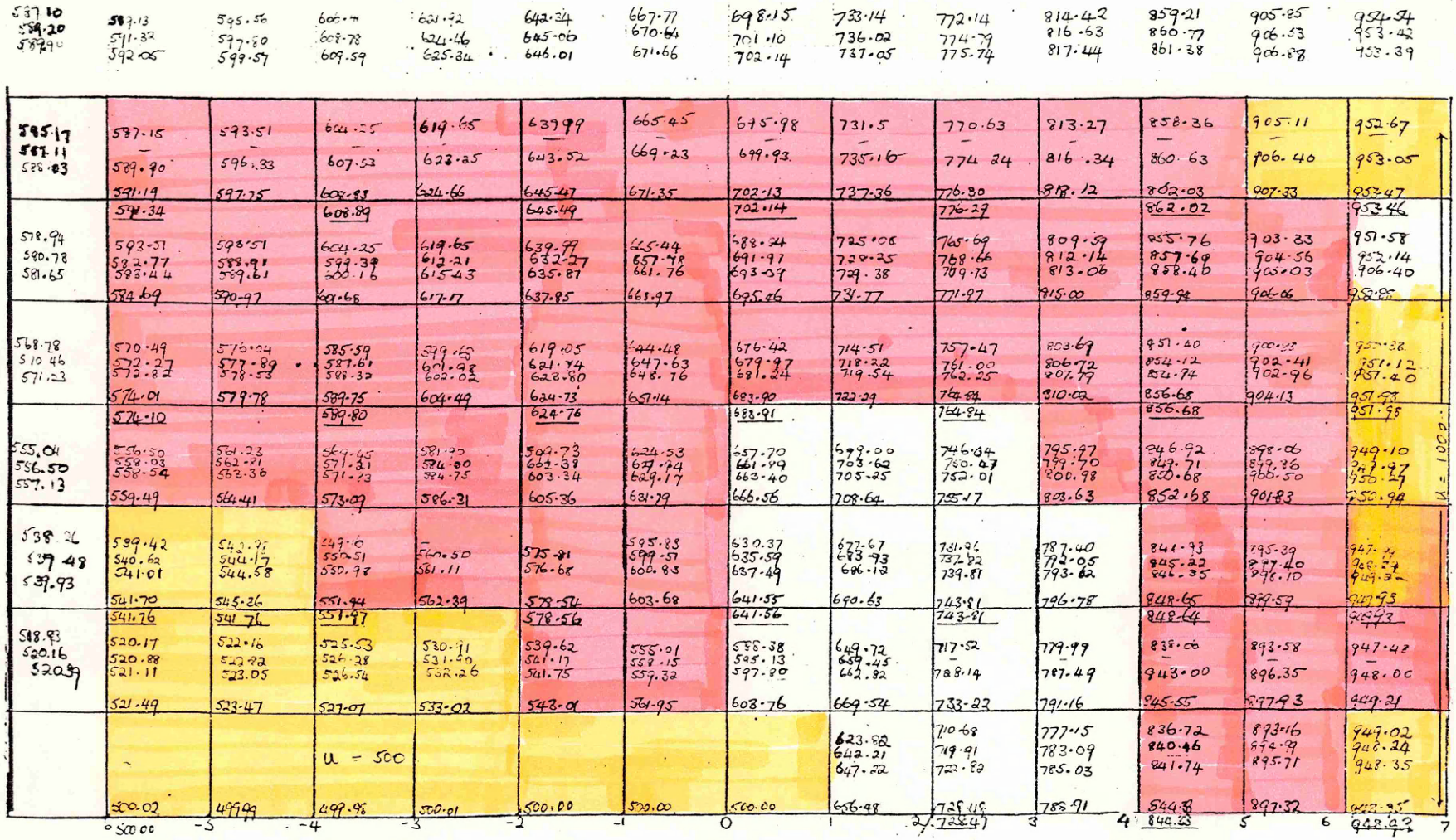


Figure 5.11

Table 5.2 Problem 5.2b solved with uniform meshes



Key

Difference $\leq$ 1

1 < Difference $\leq$ 3

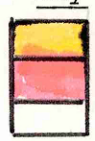
Difference $\geq$ 3

$u^{(42)}$
$u^{(84)}$ Computed values
$u^{(126)}$
Symm (96)

W + P (1977) Reference values

NB. The numbers in the square relate to lower left hand corner as in the key

Table 5.2a Problem 5.2b solved with a graded mesh



u (44)
u (51)
u (59)
u
Symm

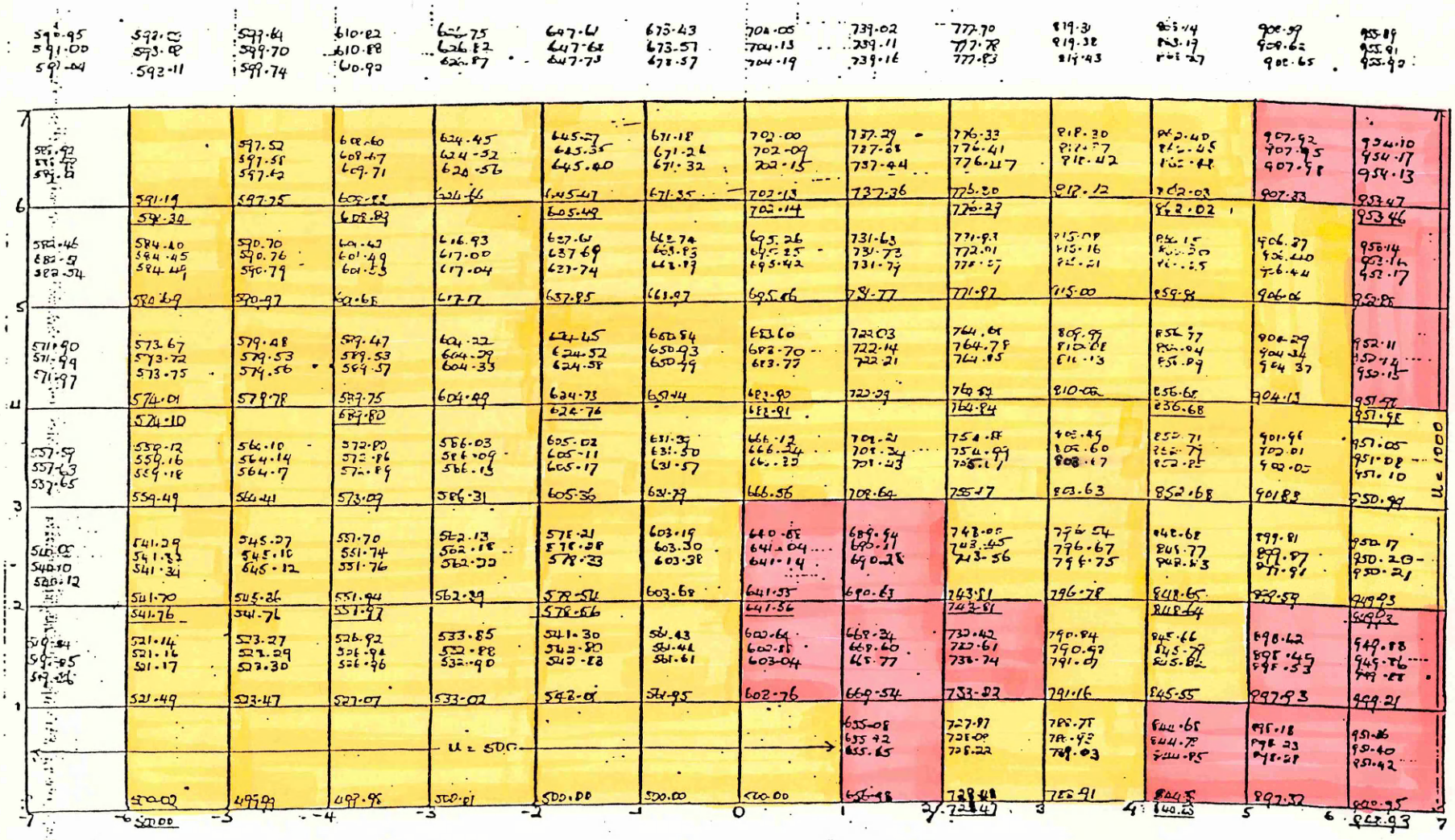
W & P

589.95	590.99	597.51	608.51	624.21	644.23	670.43	700.99	735.84	774.62	816.49	860.66	906.45	953.37
589.96	592.05	598.62	609.71	625.58	646.26	671.97	702.50	737.43	776.12	817.81	861.72	907.19	953.75
590.80	592.27	599.48	610.24	626.55	647.38	673.17	703.74	737.67	777.29	818.82	862.53	907.75	954.04
							0						
586.74		595.42	606.32	621.93	642.49	668.15	698.80	734.05	773.21	815.44	859.94	905.96	952.89
587.89		596.52	607.50	623.23	643.93	669.70	700.43	735.67	774.74	816.79	861.02	906.71	953.27
588.78	590.92	597.37	608.43	624.25	645.05	670.92	701.70	736.94	776.93	817.83	861.85	907.29	953.57
	591.19	597.75	608.83	624.66	645.47	671.35	702.13	737.36	776.30	818.12	862.03	907.33	953.47
580.62	581.34		608.89		645.49		702.14		776.29		862.02		953.46
581.58	582.53	589.23	599.23	614.45	634.80	661.87	691.87	728.16	768.57	812.06	857.63	904.51	—
582.32	583.51	589.75	600.37	615.74	636.25	662.23	693.63	729.93	770.25	813.53	858.80	905.32	—
	584.26	590.55	601.26	616.75	637.40	663.61	694.98	731.30	771.55	814.66	859.70	906.05	
	584.69	590.97	601.68	617.17	637.85	663.97	695.46	731.77	771.97	815.00	859.98	906.06	953.85
570.27	572.01	577.68	587.44	601.84	621.62	647.51	679.85	719.10	760.89	806.62	854.84	902.36	951.10
571.12	572.88	578.62	588.50	603.08	623.09	649.24	681.80	722.14	762.83	808.31	855.37	903.27	951.56
571.78	573.56	579.35	589.33	604.05	624.25	650.61	683.32	721.72	764.32	809.60	856.59	903.96	951.91
	574.01	579.78	589.75	604.49	624.73	651.14	683.90	722.29	764.84	810.02	856.68	904.13	951.98
	574.10		589.80		624.76		683.91		764.84		856.68		951.98
556.27	557.75	562.60	571.05		602.26	627.82	661.77	702.48	750.34	794.58	849.63	899.81	949.94
556.96	558.47	563.88	571.91	—	603.71	629.69	664.05	705.97	752.70	797.39	851.76	900.84	950.45
557.50	559.02	564.00	572.69	585.29	604.87	631.19	665.87	707.91	754.51	803.11	850.32	901.62	950.85
	559.49	564.41	573.09	586.31	605.36	631.79	666.56	708.64	755.17	803.63	852.68	901.83	950.94
537.14	540.32	543.98	550.37	560.38	575.70	599.39	635.45	683.75	737.64	791.90	845.12	897.34	948.86
537.63	540.83	544.55	551.06	561.30	577.01	601.40	638.38	687.07	720.66	794.32	846.90	898.50	949.43
540.01	541.28	545.00	551.62	562.03	578.08	603.02	640.65	689.64	742.95	796.14	848.24	899.38	949.87
	541.70	545.36	551.94	562.39	578.54	603.68	641.55	690.63	743.81	796.78	848.65	899.59	949.93
	541.76	541.76	551.97		578.56		641.56		743.81		848.64		949.93
519.36	520.63	522.69	526.19	531.93	541.08	588.06	595.91	659.21	725.26	785.45	841.65	895.55	949.19
519.61	520.90	523.99	526.57	532.26	541.96	559.83	599.04	664.21	729.16	788.29	843.64	896.83	948.62
519.80	521.11	523.23	526.88	532.80	542.67	561.31	602.44	668.04	732.05	790.39	845.12	897.78	949.28
	521.49	523.47	527.07	533.02	543.01	564.95	603.76	669.54	733.22	791.16	845.55	897.93	949.21
								642.18	719.89	783.08	840.46	840.46	948.25
								649.65	724.32	786.11	842.54	896.32	948.89
≤				u = 500				658.94	727.57	788.32	844.06	897.29	949.35
	500.02	499.99	499.98	500.01	500.00	500.00	500.00	656.48	725.19	788.91	844.38	897.32	948.95
									725.47		846.28		948.93

-6 500.00 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7

u = 1000

Table 5.3 Problem 5.2b calculated with concentrated meshes



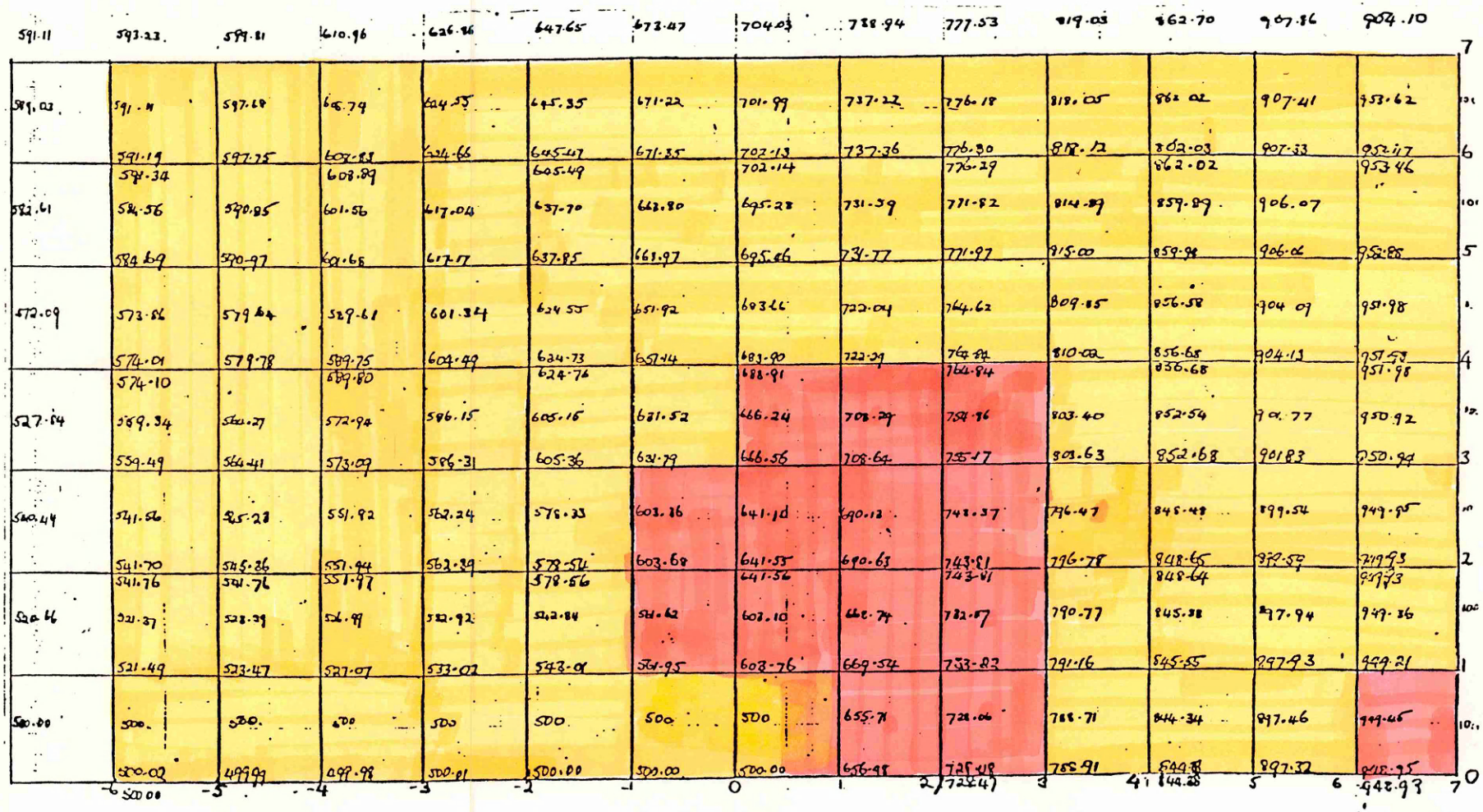
Key

Difference ≤ .3

Difference > .3

u(72)
u(80)
u(88)
u
symn

Table 5.4 Problem 5.2b calculated with graded mesh and extra grading at corners



Key

Difference ≤ .3

Difference > .3

u(78)

Symm

P & W

The meshes in Figure 5.11 were used and the results tabulated in Table 5.2(a). There is an improvement over the whole region, the results being at least as good for 59 nodes as for 126 on a uniform mesh and with an even bigger improvement near the origin where the maximum difference is .24% from the standard results.

Concentrated Mesh

Adding the extra nodes so that the extra elements in the interval about the origin are equal lengths does not improve the convergence rate so rapidly as did the graded mesh. When there were 88 nodes, i.e. the intervals each side of the origin were divided into 24 parts, the accuracy of the two methods was about the same, the maximum difference from the standard results being .21%. The results obtained with this kind of mesh are given in Table 5.3.

Corners

It was noticed that there was some deterioration in the quality of the solution in the neighbourhood of the corners, particularly where boundary conditions changed. This was improved by adding an extra node near each corner and with 78 nodes (graded about the origin). The solution shown in Table 5.4 differs from Symm's by no more than .12% anywhere.

Two more solutions were found by halving the basic element size over the whole boundary and these two solutions 118 nodes with graded mesh and 112 with concentrated mesh are compared with the 78 node solution in Table 5.5. Neither mesh produced a significant improvement.

Table 5.5 Problem 5.2b solved with basic half unit meshes. Solution $u^{(118)}$ compared with $u^{(78)}$

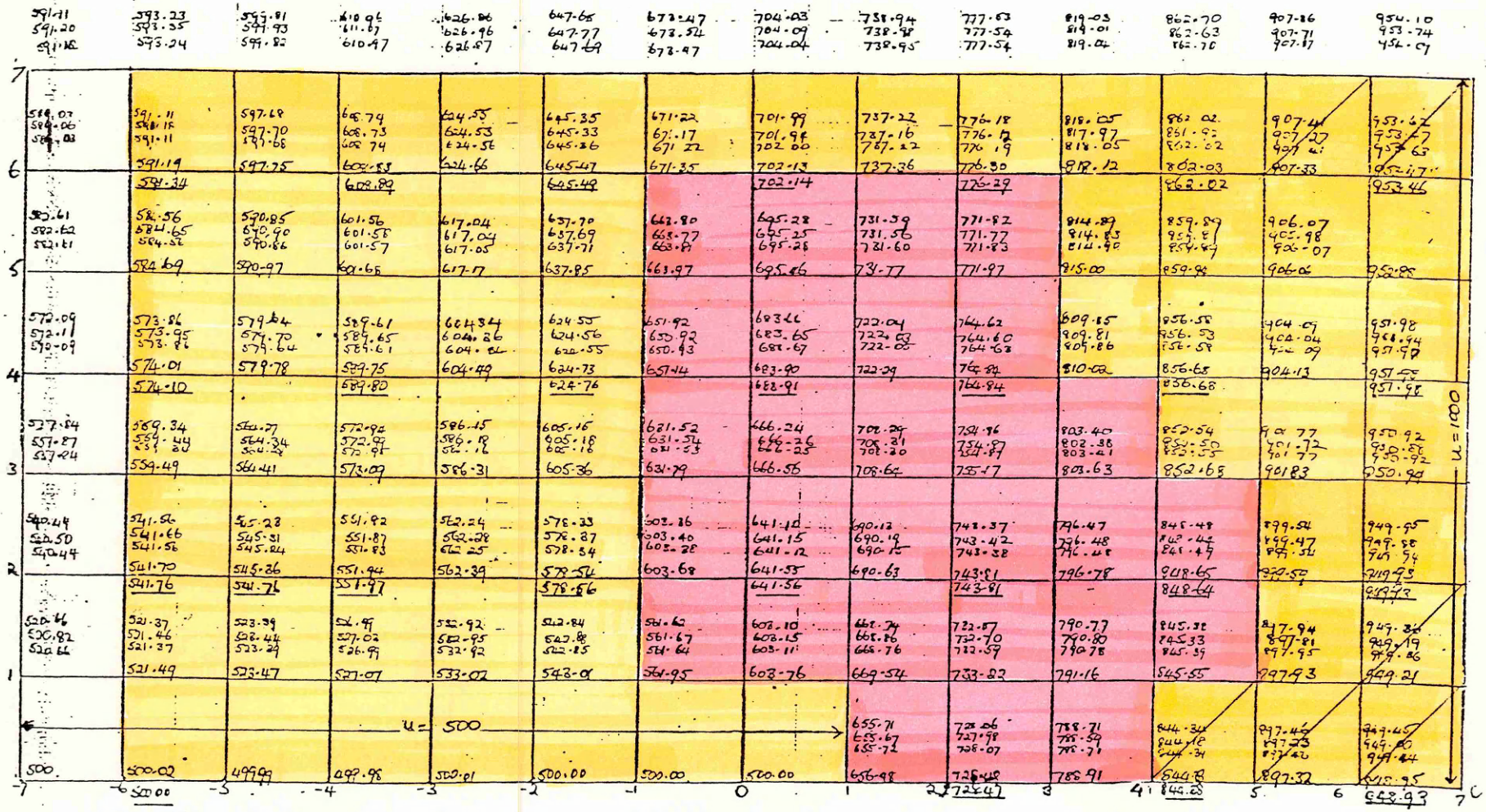
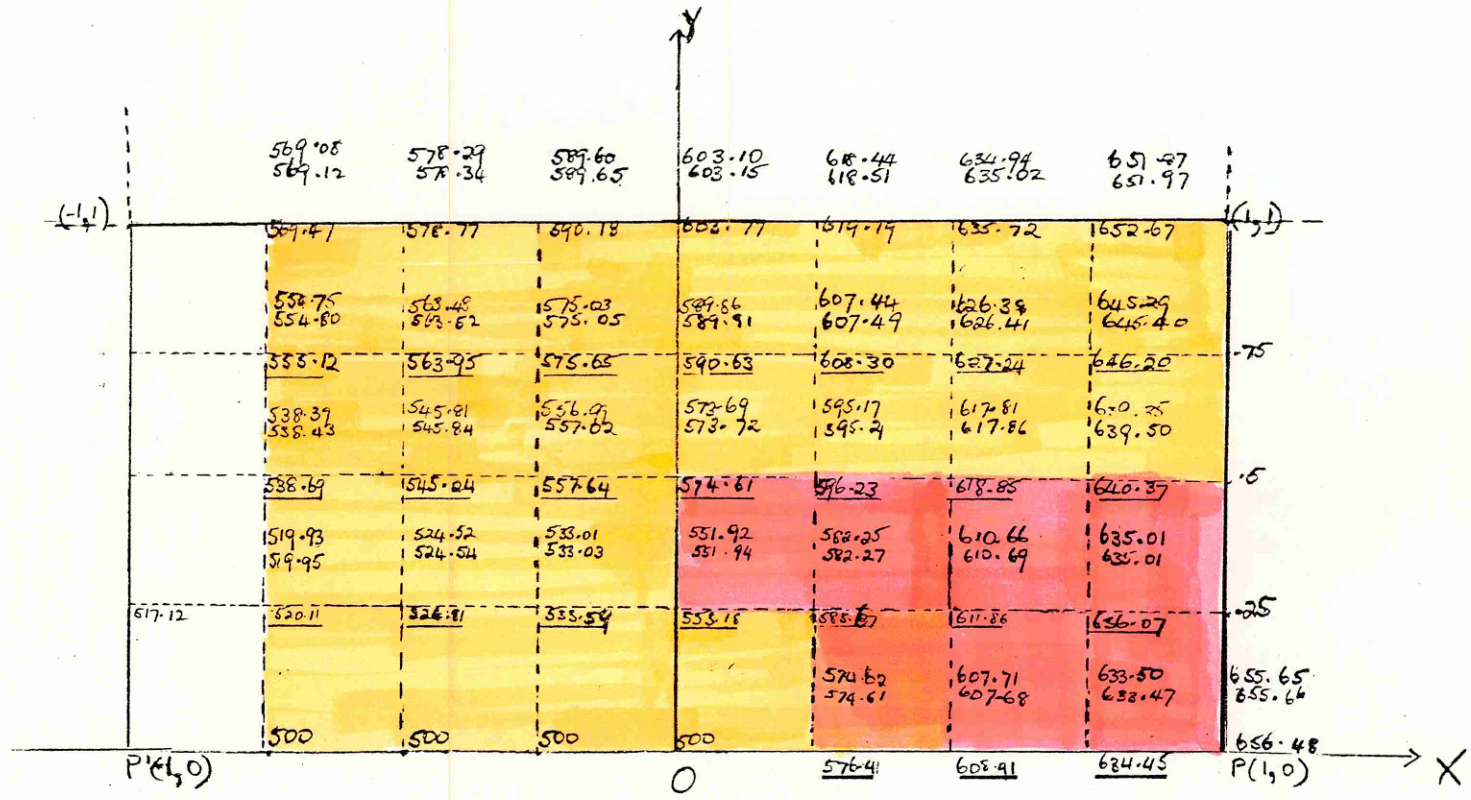


Table 5.6 Solution to problem 5.2b in the neighbourhood of O



Key

Difference ≤ 1

Difference > 1

u (78)
u (118)
Symm

Neighbourhood of the Origin

In the immediate neighbourhood of the origin, $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$, there is a maximum difference of .3% from the standard results shown in Table 5.6. The error is much higher here than elsewhere so it would seem necessary to use Symm's method if results at such points are wanted accurately.

Note: As Symm used exact integration to obtain his results some of the difference can be attributed to using numerical integration here.

Problem 5.3

To solve Laplace's equation, $\nabla^2 u = 0$, over the L-shaped region shown in Figure 5.12 with boundary conditions as specified.

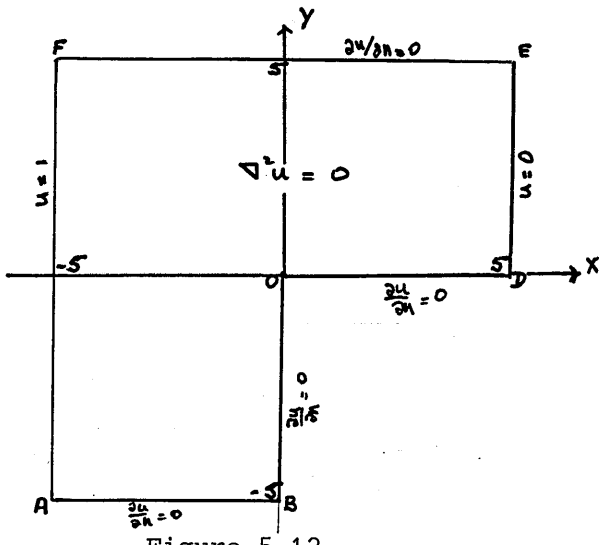


Figure 5.12

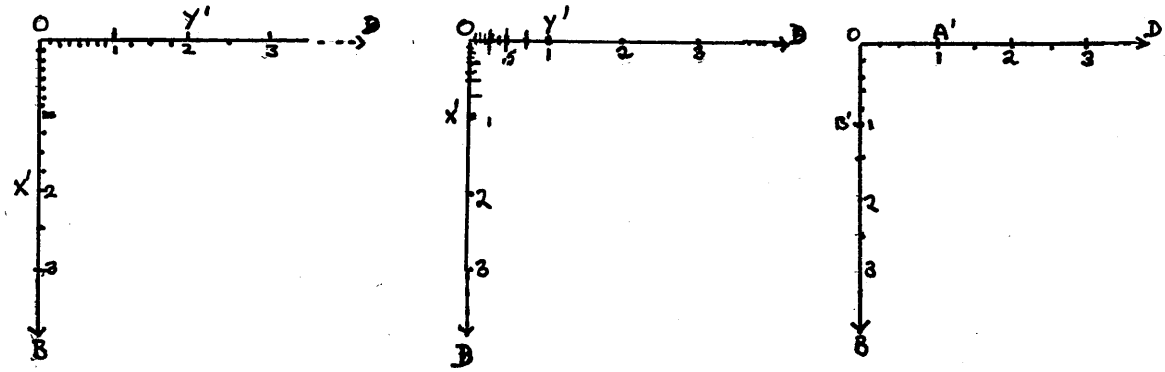
This example has a re-entrant corner BOD and corners where the boundary conditions change. It has been computed by Symm[36], whose results agree with those he quotes from Papamichael and Whiteman[38]. The results obtained here are compared with Symm's.

Results

Uniform Mesh

The results with a series of such meshes (40, 80, 120 nodes) are shown in Table 5.7. It can be seen that the biggest difference from the standard results is along the boundary BOD where the two worst regions are near the corner O (not at it) and the corner D. The maximum difference between the solutions is .5% with 120 nodes, an improvement from .85% with 80 nodes.

Mesh refined near O

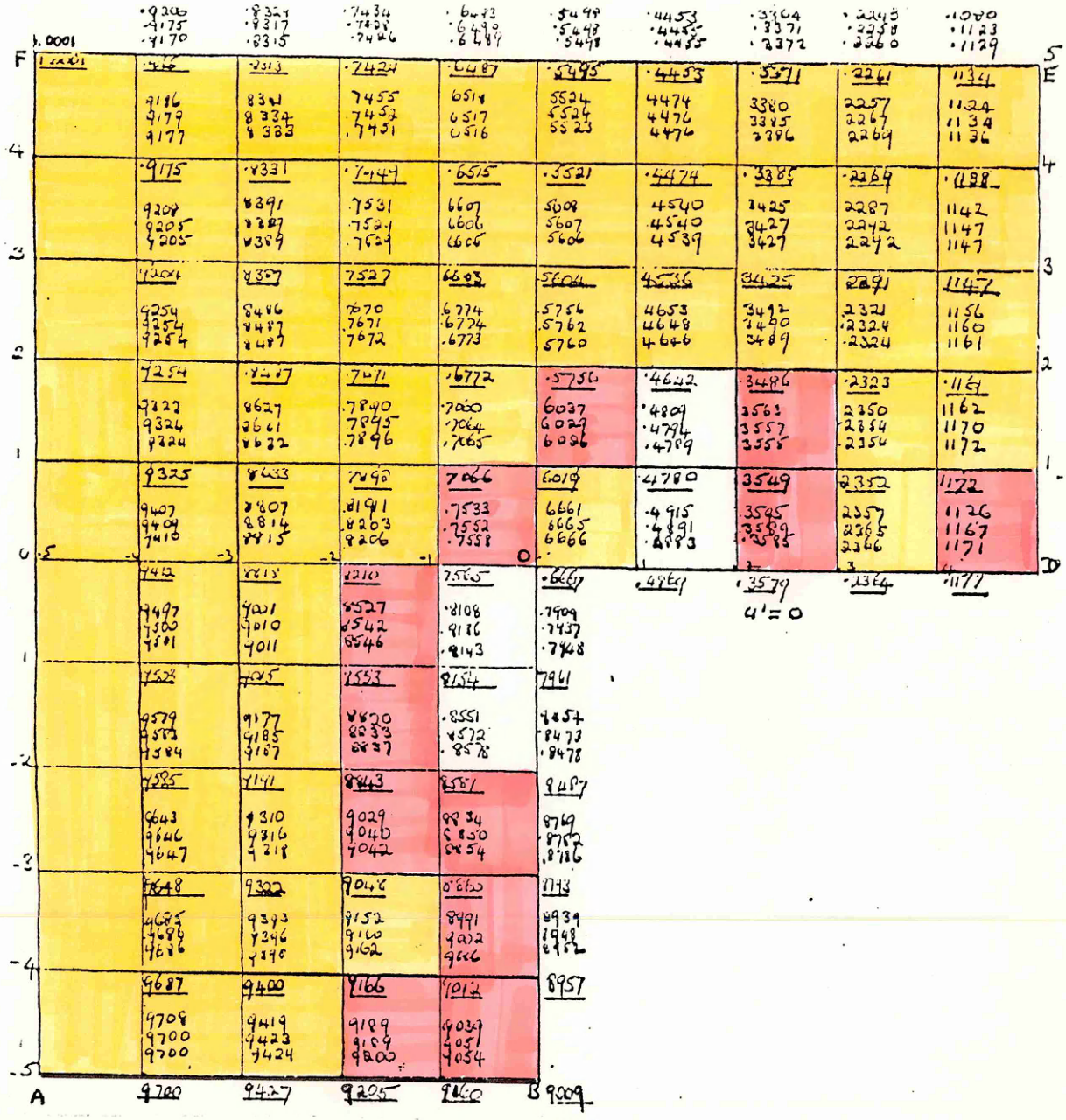


66 nodes 58 nodes 54 nodes

Figure 5.13

The boundary was divided into elements of unit length except along BOD. Here half unit length elements were used with extra refinements added near O. The extra nodes were placed in three different ways - as graded meshes between X'O and OY' (Figures 5.13(a)

Table 5.7 Problem 5.3 solved with uniform meshes



Key

Difference ≤ .0004

.0004 < Difference ≤ .0007

Difference > .0007

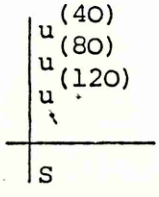


Table 5.8 Problem 5.3 solved with graded mesh around O
(as shown in Figure 5.13)

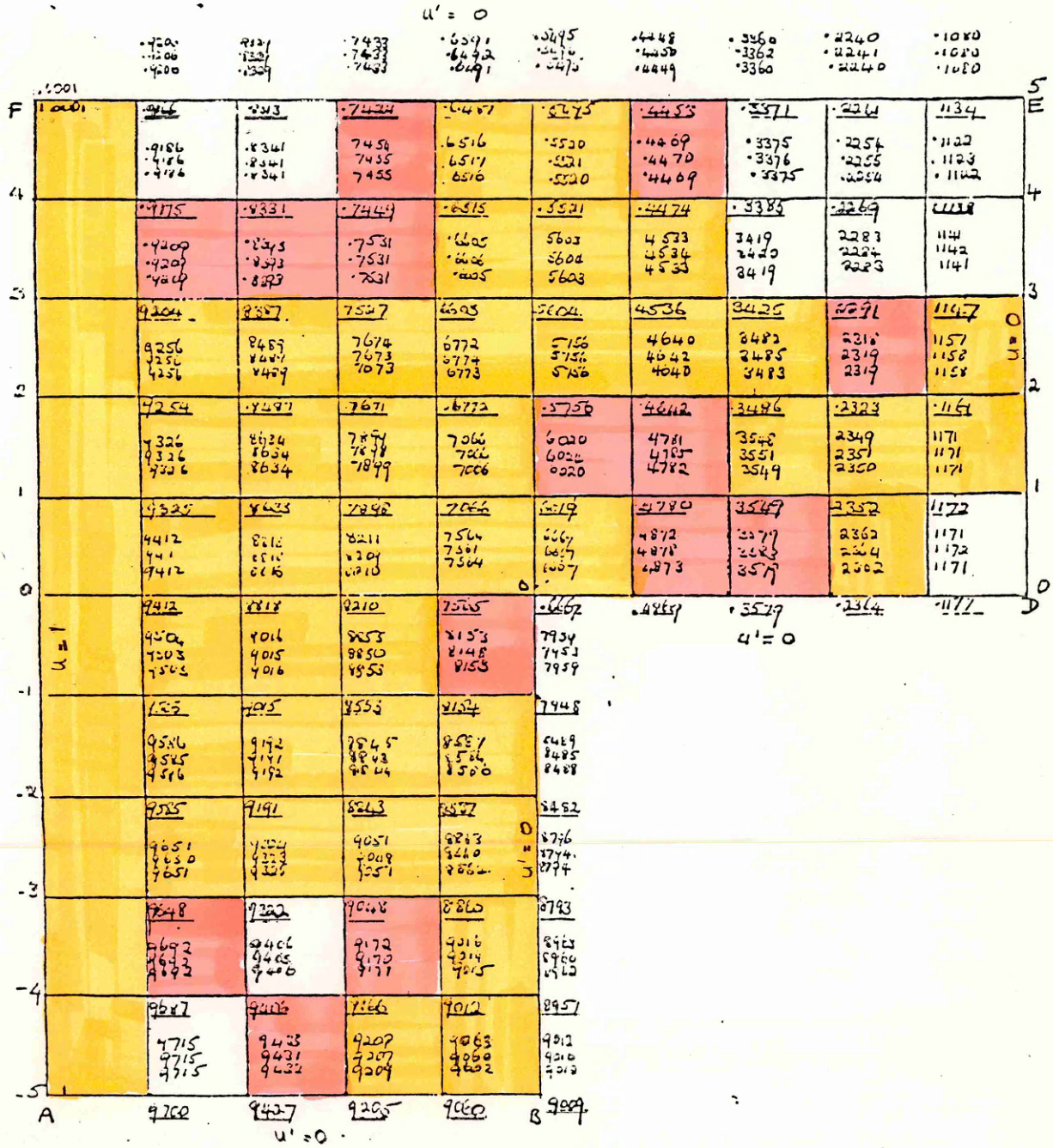
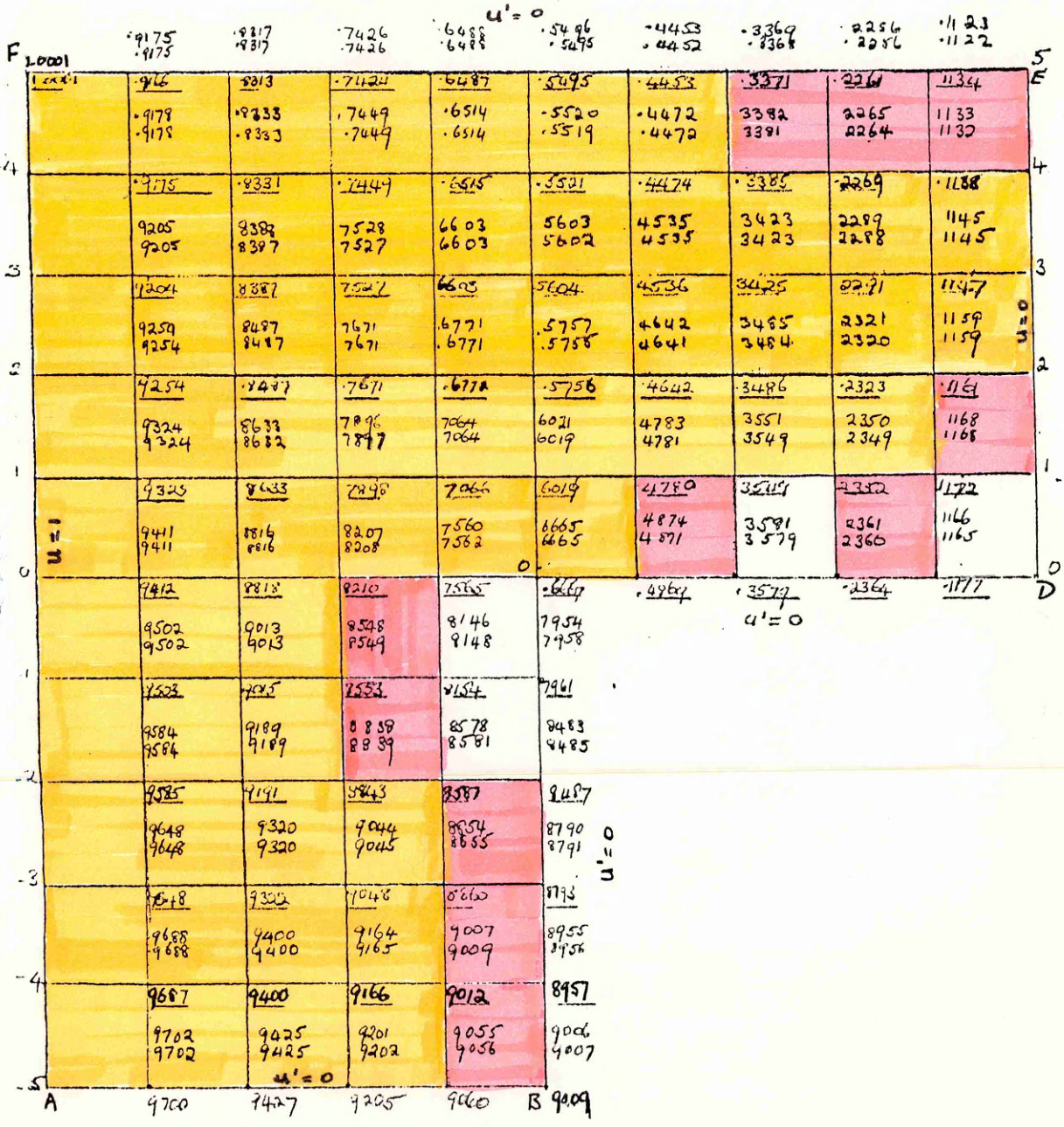


Table 5.9 Problem 5.3 solved with meshes with extra refinement at all corners



Key

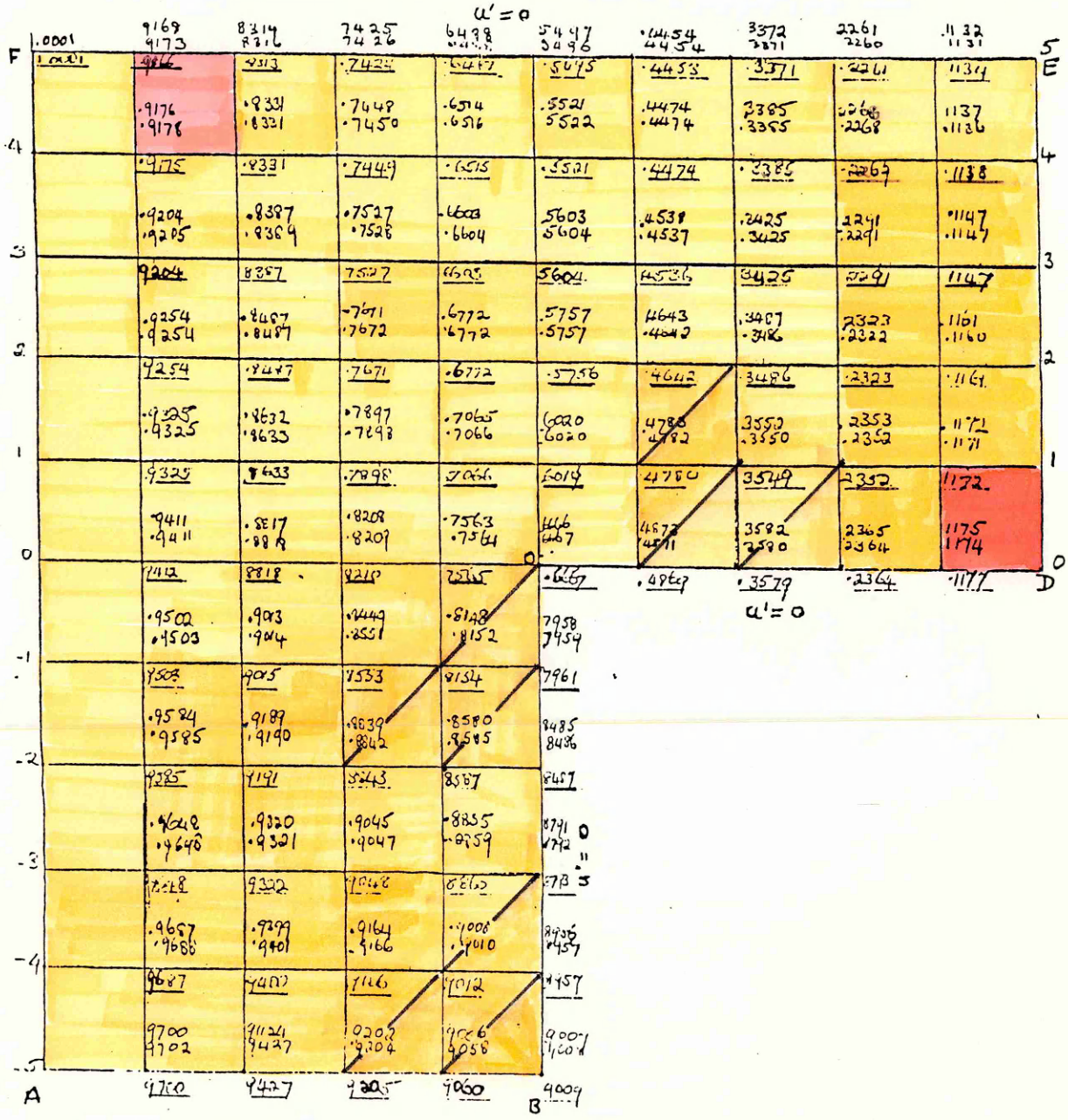
Difference ≤ .0004

.0004 < Difference ≤ .0006

Difference > .0006

u (60)
u (76)
S

Table 5.10 Problem 5.3 solved using graded mesh at O and extra corner refinement at D & F



and (b)) or as concentrated meshes between BO and OA' (Figure 5.13(c)). The solutions computed with these meshes show an improvement near the BOD boundary with the regions near the corners A, E+D showing the biggest difference from Symm's results) (at E the difference is 1.4%). The solution compares well with that calculated with a uniform mesh of 120 nodes in Table 5.7.

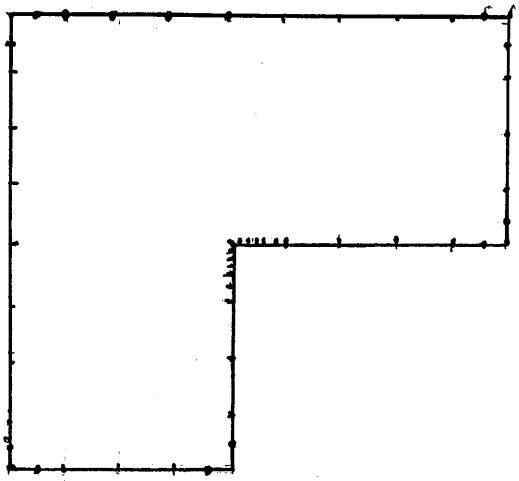


Figure 5.14

The results in Table 5.8 suggest that some extra nodes should be added near the corners. The boundary was first subdivided into equal length elements refined as before at the origin and an extra node added in elements adjacent to each corner (see Figure 5.14). The results for two meshes (nodes 60 and 76) are shown in Table 5.9 and show that apart from corners D and E where it is .7%, the maximum difference has improved to .07%. Further refinement (86 nodes) was done at D and E and the results are in Table 5.10. The program did not allow further refinement at the origin. Instead, the basic half unit mesh was kept and refinements at the O, D and E, as shown on Figure 5.15, giving 100 nodes - now a maximum difference overall of at least .26% was achieved.

(Figures 5.14 and 5.15 are not to scale).

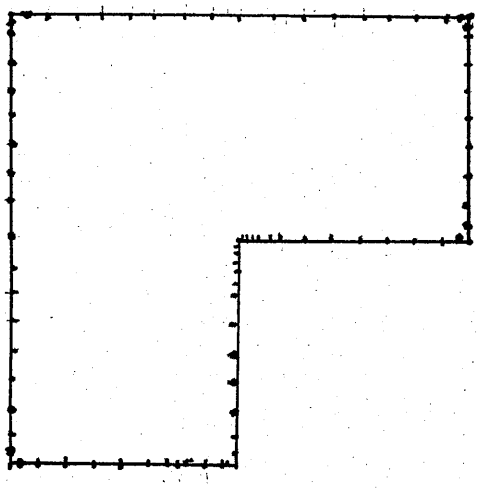


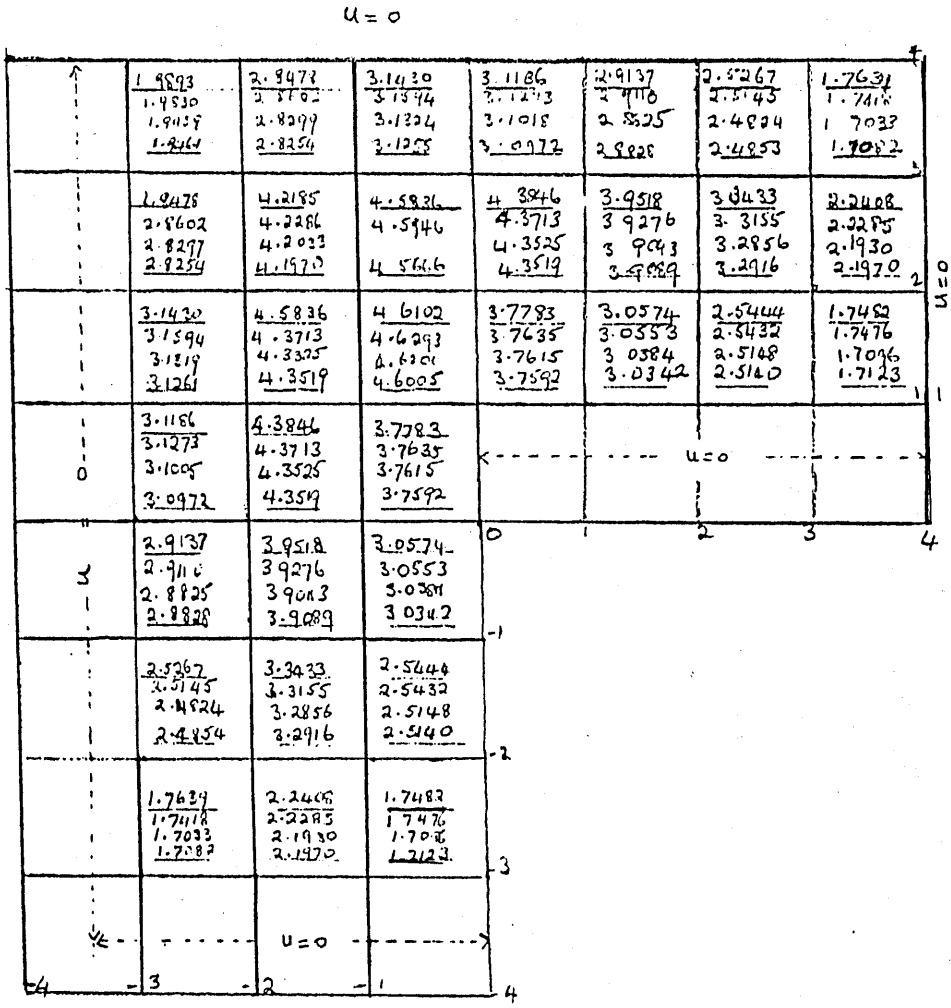
Figure 5.15

Table 5.11 summarises these results showing how the position of the maximum difference from the standard solution varied.

Table 5.11 Summary of results for Problem 5.3

Nodes	Mesh	Maximum %	Difference Position	Maxima Difference % Near O	Solution values u(-1,-1)	u(1,0)
40	Uniform	1.10	0	1.10	.8108	.4915
80	Uniform	.45	0	.45	.8136	.4891
120	Uniform	.3	0	.3	.8143	.4883
54	Graded	1.40	E	.13	.8153	.4878
58	Graded	1.40	E	.06	.8148	.4873
66	Graded	1.40	E	.13	.8153	.4872
76	Graded & Corners	.7	D	.07	.8148	.4871
86	Graded	.16	E	.08	.8148	.4873
100	Graded	.26	E	.02	.8153	.4871
Standard	(Symm)				.8154	.4869

Table 5.12 Comparison of results obtained by Finite Element Method and the Boundary Element Method when solving Poisson's Equation, $\nabla^2 u = -2$ satisfying the boundary condition $u = 0$ on L shaped region



Key

$u^{(133)}$	FEM
$u^{(36)}$	BEM
$u^{(60)}$	BEM
$u^{(341)}$	FEM

Conclusion

The adding of extra nodes has been an effective and simple way of improving results near singularities. The approach has been experimental - extra nodes being positioned where the difference from the standard solution was greatest. It proved necessary to do most refinement in the neighbourhood of re-entrant corners, but some was needed at corners, particularly where the boundary condition changed. Any number of singularities can easily be handled as it is only the discretization of the boundary that is affected and no program changes are needed.

Comparison

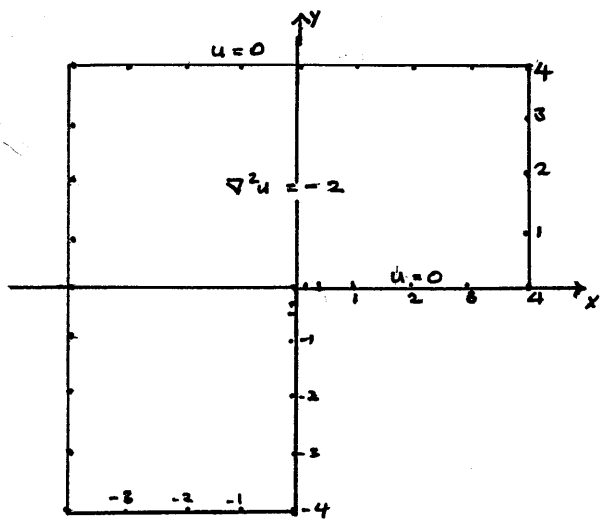


Figure 5.16

The solutions obtained to problem 5.1 using both methods are compared in Table 5.12. The finite element solutions were obtained using the element patterns shown in Figures 5.4(b) and (c), while the boundary element method used a graded refinement around the re-entrant corner as shown in Figure 5.16. This mesh corresponds to the one giving .06% difference from Symm's results near the origin (m = 76 on Table 5.11).

The results of the two methods agree to within .5% sufficiently close to increase confidence in both.

CHAPTER 6

TORSION OF HOMOGENEOUS ISOTROPIC CYLINDERS

6.1 Introduction

St. Venant [40] developed a theory of torsion calculating the effect of applying pure twist to a long cylinder in the absence of body forces. The theory relies on the assumption that if some distribution of forces acting on a portion of a body is replaced by a different but statically equivalent set of forces acting on the same portion of the body, then the two distributions of forces produce the same effect at sufficiently great distances from the area of action. It is therefore necessary that the length of the cylinder should be much greater than the diameter of the cross section. He gave the first solution to the problem of the torsion of a cylinder by a couple applied at the ends. His method was to make assumptions regarding the deformations of the twisted bar and then to show with these assumptions he could satisfy the equations of equilibrium and the boundary conditions. As the elasticity equations have been shown to have a unique solution [42] Saint Venant had thus obtained the solution.

6.2 Warping Function

St. Venant assumed that the deformation of the twisted cylinder would take the form of a rotation of the cross section combined with a warping of each cross section that is the same for all cross sections. Taking the origin at one end and the z axis, parallel to the length of the cylinder as in Figure 6.1

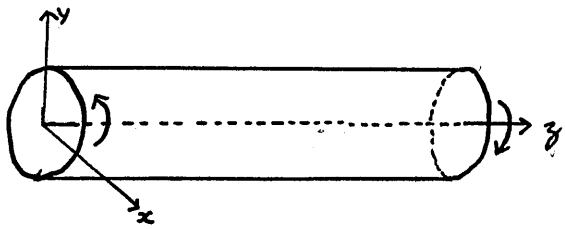


Figure 6.1

these deformations take the form,

$$u = -\theta yz$$
$$v = \theta zx \qquad \ldots\ldots (6.1)$$
$$w = \theta\psi(x,y)$$

where u, v, w are the displacements in the x,y,z directions, respectively, and θ is the twist and ψ the warping function, showing that with these assumptions the equations of equilibrium [43] describing the state of stress in the body are satisfied together with the boundary conditions. The stress components in an isotropic body have also, in general, to satisfy the compatibility conditions [44] before they can be fully determined.

The only non-vanishing strain components [45] are

$$e_{zx} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = \theta\left(\frac{\partial \psi}{\partial x} - y\right)$$

and $\qquad \ldots\ldots (6.2)$

$$e_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} = \theta\left(\frac{\partial \psi}{\partial y} + x\right)$$

The corresponding stress components are found using Hooke's Law

$$\tau_{zx} = G\theta\left(\frac{\partial\psi}{\partial x} - y\right)$$

$$\tau_{yz} = G\theta\left(\frac{\partial\psi}{\partial y} + x\right) \qquad \ldots\ldots (6.3)$$

where G, the modulus of rigidity, is a constant for isotropic materials.

Substituting these stress components in the equation of equilibrium the function ψ is found to satisfy

$$\left(\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2}\right) = 0 \qquad \ldots\ldots (6.4)$$

while the boundary conditions reduce to

$$\tau_{xz}\cos(n,x) + \tau_{yz}(\cos n,y) = 0 \qquad \ldots\ldots (6.5)$$

where cos(n,x) and cos(n,y) are the direction cosines of the outward normal. If s is the arc length (increasing from A to B) then

$$\cos(n,x) = \frac{dy}{ds}$$

$$\cos(n,y) = -\frac{dx}{ds} \qquad \ldots\ldots (6.6)$$

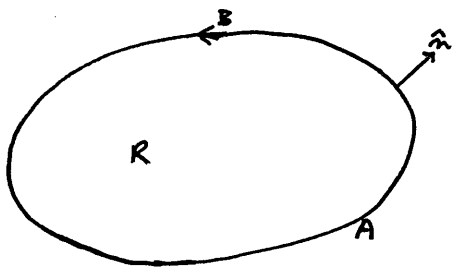


Figure 6.2

and equation (6.5) after substituting for the stress components (6.3) becomes

$$(\frac{\partial\psi}{\partial x} - y)\ \frac{dy}{ds} - (\frac{\partial\psi}{\partial y} + x)\ \frac{dx}{ds} = 0 \qquad \ldots\ldots (6.7)$$

$$\frac{d\psi}{dn} = \frac{1}{2}\frac{d}{ds}(x^2 + y^2) = \frac{1}{2}\frac{d}{ds}r^2 \qquad \ldots\ldots (6.8)$$

Solving the torsion problem using the warping function is thus equivalent to solving a Neumann boundary value problem defined by the Laplace equation (6.4) together with boundary condition (6.8).

6.3 Stress Function

The equations of equilibrium[43] reduce to

$$\frac{\partial}{\partial z}\tau_{xz} = 0,\ \frac{\partial}{\partial z}\tau_{yz} = 0$$

and $$\frac{\partial}{\partial x}\tau_{xz} + \frac{\partial}{\partial y}\tau_{yz} = 0 \qquad \ldots\ldots (6.9)$$

The first two of these equations are already satisfied by the stress components in (6.3), and, to satisfy the third, a function ϕ, called the stress function, is defined so that

$$\tau_{xz} = \frac{\partial\phi}{\partial y} \quad \text{and} \quad \tau_{yz} = -\frac{\partial\phi}{\partial x} \qquad \ldots\ldots (6.10)$$

Then

$$\frac{\partial \phi}{\partial y} = G\theta \left(\frac{\partial \psi}{\partial x} - y\right)$$

$$\frac{\partial \phi}{\partial x} = -G\theta \left(\frac{\partial \psi}{\partial y} + x\right) \qquad \ldots\ldots (6.11)$$

and eliminating ψ, the stress function ϕ satisfied Poisson's equation

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = -2G\theta \qquad \ldots\ldots (6.12)$$

The boundary condition becomes, in terms of ϕ,

$$\frac{\partial \phi}{\partial y}\frac{dy}{ds} + \frac{\partial \phi}{\partial x}\frac{dx}{ds} = \frac{d\phi}{ds} = 0 \qquad \ldots\ldots (6.13)$$

which is equivalent to ϕ being constant along the boundary of the cross section. In the case of singly connected cross sections, this constant can be chosen arbitrarily and is usually taken as zero, i.e. the boundary condition is

$$\phi = 0 \qquad \ldots\ldots (6.14)$$

Solving the torsion problem using the stress function is equivalent to solving the boundary value problem defined by Poisson's equation (6.12) together with the boundary condition (6.14).

6.4 Rigidity

To show that the stresses calculated give rise to forces at the end statically equivalent to a couple with axis parallel to that of the cylinder, it is necessary to show that

$$\iint \tau_{xz}\, dx\, dy = 0 \quad \text{and} \quad \iint \tau_{yz}\, dx\, dy = 0 \,. \qquad \ldots\ldots (6.15)$$

Using the stress function ϕ, defined in equations (6.10) gives

$$\iint \tau_{xz}\, dx\, dy = \iint \frac{\partial \phi}{\partial y}\, dx\, dy = \int dx \int \frac{\partial \phi}{\partial y}\, dy = 0 \qquad \ldots\ldots$$

and similarly

$$\iint \tau_{yz}\, dx\, dy = 0 \,.$$

These forces represent a couple of magnitude, M, called torsional rigidity, where

$$M = \iint (x\, \tau_{yz} - y\, \tau_{xz})\, dx\, dy \,. \qquad \ldots\ldots (6.16)$$

It can be expressed in terms of the stress function by

$$M = -\iint (x\, \phi_x + y\, \phi_y)\, dx\, dy \qquad \ldots\ldots (6.17)$$

and integrated to give

$$M = \iint - \{(x\phi)_x + (y\phi)_y + 2\phi\}\, dx\, dy \qquad \ldots\ldots (6.18)$$

$$= 2 \iint_R \phi\, dx\, dy - \int_B \phi(y \cos(n,x) + m \cos(n,y))\, dS \qquad \ldots\ldots (6.19)$$

$$= 2 \iint \phi\, dx\, dy \text{ (when } \phi = 0 \text{ on the boundary)} \qquad \ldots\ldots (6.20)$$

Alternatively, using the expressions for the stress components in equations (6.3) the rigidity M may be rewritten as

$$M = G\theta \left(\iint_R x(\psi_y + x) - y(\psi_x - y)\right) dA \qquad \ldots\ldots (6.21)$$

$$= G\theta \left(\iint_R (x^2 + y^2) + (x\,\psi_y - y\,\psi_x)\right) dA \qquad \ldots\ldots (6.22)$$

Using Green's theorem this becomes

$$M = G\theta \left(\iint_R r^2\, dA - \int_B \left(x\psi \frac{dx}{ds} + y\psi \frac{dy}{ds}\right) dS\right) \qquad \ldots\ldots (6.23)$$

and applying the boundary condition (6.7) gives

$$M = G\theta \left(\iint_R r^2\, dA + \int_B \psi \frac{d\psi}{dn}\, dS\right) . \qquad \ldots\ldots (6.24)$$

6.5 Solution by the Finite Element Method

The stress function formulation is particularly suitable for use with the finite element method. It results in the values of ϕ at mesh points throughout the region allowing the rigidity to be conveniently calculated using these values and the finite element mesh.

This solution method can conveniently be extended to cover non-homogeneous, anisotropic materials [48].

Program Modifications to Finite Element Method

The calculation routines of the standard finite element method need no changes to solve the stress problem, but an amendment is necessary to calculate the rigidity, most conveniently placed in the output routines. The values of ϕ at nodal parts are used to find the integral (6.20), performing the numerical integration by the same procedure used in the main calculation. The element mesh is used as the mesh for the numerical integration.

6.6 Solution by the Boundary Element Method

The warping function can be conveniently found numerically using the boundary element method. This was first done by Jaswon and Ponter [47] who use the formula (6.24) for finding the torsional rigidity from the values of the warping function and its normal derivatives on the boundary, an approach that avoids the need to calculate values throughout the region.

Program Modifications to Boundary Element Method

As only cross sections bounded by straight lines were being considered, the boundary condition

$$\frac{d\psi}{dn} = + \frac{d}{ds} \left(\frac{1}{2} r^2\right)$$

was implemented by program and the integral around the boundary in

formula (6.24) calculated numerically using the discretization already set up when positioning the elements. The polar moment of inertia, $\iint r^2 \, dA$, was calculated by subdividing the cross section into rectangles and calculating the contribution from each.

Note For simplicity in the calculation the constant $G\theta$ was taken as 1 for all calculations.

Problem 6.1

To solve the torsion problem over the square ABCD, $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$ shown in Figure 6.3

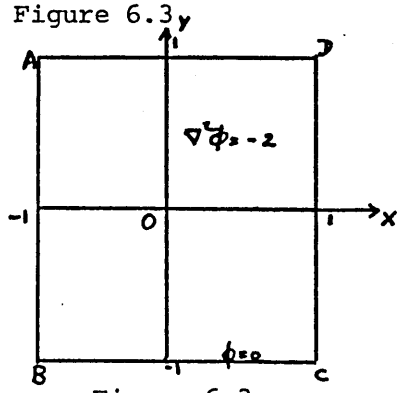


Figure 6.3

Stress Function

The stress function was found by solving Poisson's equation, $\nabla^2\phi = -2$, over the square satisfying $\phi = 0$ on the boundary using the finite element method (Problem 3.1).

Results

Table 6.1 Torsion of a square section using finite element method

Elements	$\phi(0,0)$	Rigidity (M_S)	Error % in M
16	.6214	2.0464	9.03
64	.5968	2.1972	2.32
256	.5898	2.2480	.07
Analytic	.5894	2.2496	

Warping Function

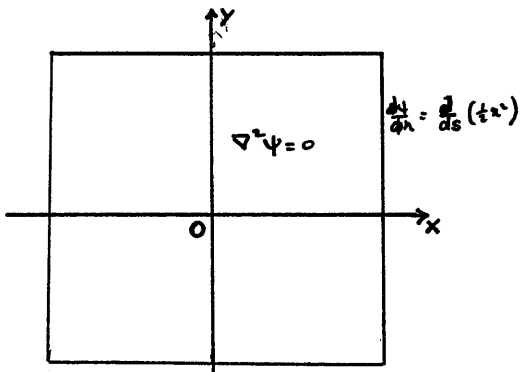


Figure 6.4

The warping function was found over the square section illustrated in Figure 6.4 by solving the boundary value problem in Section 6.2 using the boundary element method. The whole boundary was divided into elements of equal size and linear interpolation used over each element. As it is a Neumann problem it was necessary to specify the value of the warping function at one point at least. Symmetry requires that it should be zero at the ends of the axes of symmetry and these values were used. Just taking it zero at one such point did not prove to be sufficient for the solution to reflect the symmetry of the problem.

Results

Table 6.2 Torsion of a square section using boundary element method

Elements	Rigidity (M)	Error % (M_s)	Max. Stress
16	2.5552	13.58	1.2230
32	2.3725	5.46	1.3294
48	2.3135	2.24	1.3421
64	2.2889	1.75	1.3461
Analytic	2.2496		1.351

Note The values of the torsional rigidity obtained from the two formulations give bounds for the true value, i.e.

$$(2.2480) \lesssim 2.2496 \lesssim (2.2889)$$

Torsion of a rectangle

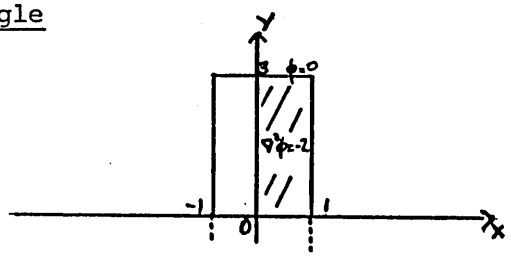


Figure 6.5

Stress Function

The finite element method was used to find the stress function by solving the boundary value problem, illustrated in Figure 6.5, over a quarter rectangle and extending to the whole domain using symmetry. It was found that while 1 point Gaussian quadrature was accurate for finding the stress function, it was necessary to use a higher order integration (3-point) when evaluating the integral for the rigidity.

Results

Table 6.3 Torsion of rectangular section by finite element method

Elts	Stress at Centre	Rigidity (M)	Error % in M_s
48	.9826	12.4348	1.52%
225	.9817	12.5880	.29%
Analytic		12.624	

Warping Function

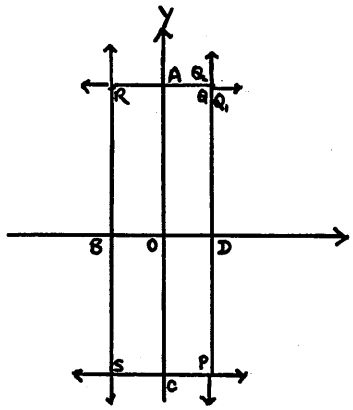


Figure 6.6

The warping function was calculated by the boundary element method. The boundary was divided into elements of equal size and the arbitrary constant of the Neumann problem set by specifying the warping function to be zero at points A,B,C,D on the axes.

In this example it was necessary to find some way of specifying the value of the normal derivative at the corners P, Q, R and S where it changes direction. The boundary element method as implemented gave poor results for the rigidity and the symmetry of the problem was not reflected in the results. Various ways of dealing with this problem have been developed. Brebbia and Dominguez [29] use two nodes very near the corner but on different sides of the corner, e.g. at Q in the rectangle in Figure 6.6. The two nodes Q_1 and Q_2

are used. This method was used here as it could be handled conveniently by the program without amendment. The extra nodes at the corners P, Q, R and S were placed so as to ensure symmetry in the solution. The calculated warping pattern agreed with the theoretical one[46]. To improve the accuracy of the solution, the symmetry of the problem was used and the warping function found on the quarter of the rectangle with boundary conditions as shown in Figure 6.7.

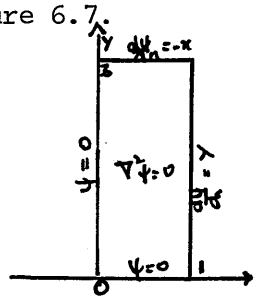


Figure 6.7

Results

Table 6.4 Torsion of rectangular section by boundary element method

Elts	Rigidity (M_w)	Max. Shear	Error % in M_w
Whole Rectangle			
52	12.7957	2.1642	1.35%
68	12.6423	2.6423	.14%
100	12.6359	2.1580	.12%
Using Quarter Rectangle			
33	12.8512	1.9941	1.8%
65	12.6928	1.9971	.5%
Analytic	12.624	1.970	

The two methods give bounds for the torsional rigidity, i.e.

$$(12.588)_s \leq 12.624 \leq (12.636)_w$$

Torsion over L-shaped Region

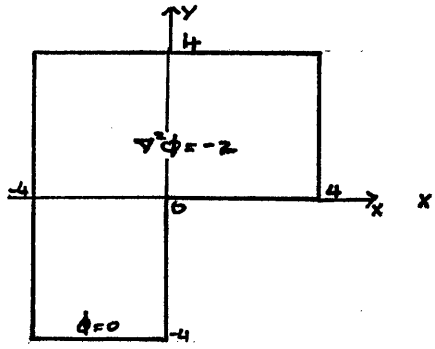


Figure 6.8

Stress Function

To find the stress function the boundary value problem illustrated in Figure 6.8 was solved by the finite element method using the meshes from Section 5.2.

Results

Table 6.5 Torsion of an L section using finite element method

Elements	Mesh	Rigidity (M_s)
48	a	203.8007
192	a	214.8689
147	b	208.8345
133	c	208.5430
341	c	216.1063

Comment

Although the even mesh gives poor results near the re-entrant corner (Section 5.2) it gave a reasonable estimate of the rigidity.

Warping Function

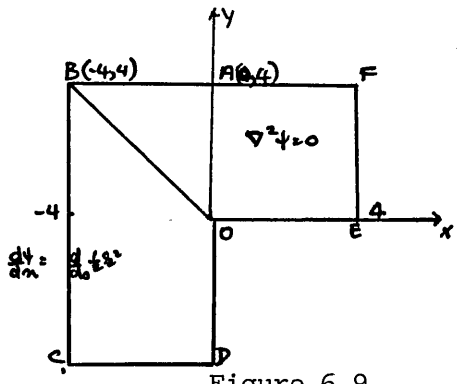


Figure 6.9

To find the warping function for an L section the boundary value problem in Section 6.8 was solved by the boundary element method. A graded mesh, as used on Problem 5.3, was used together with double nodes at the corners to cope with the change in direction of the normal at such points. The particular points at which the warping function is set to zero were O(0,0) and B(-4,4) on the line of symmetry for this section. The polar moment of inertia was calculated by dividing the region into rectangles.

Results

Table 6.6 Torsion of an L section by the boundary element method

Elements	Rigidity (M_w)	Max. Shear
62	230.770	15.5265
92	222.643	15.59

For this shape there is no theoretical result but by using the results from the two methods bounds can be obtained for it, i.e.

$$(216.1)_s \lesssim R \lesssim (222.64)_w$$

which gives the result to within .3 %. These bounds could be improved by calculation with a finer mesh in each case.

Note The calculation of the torsional properties of an L-shaped cylinder shows the difficulties that are going to be met when approaching it via the stress function or the warping function on this kind of cross section.

The use of the stress function is the most straightforward introducing no further complications than have already been met in the use of both methods as discussed earlier.

The warping function formulation resulted in a Neumann boundary value problem for which it is necessary to know the warping at one point to set the arbitrary constant arising in the solution. In the problems worked here symmetry was used to do this, a technique which is only of restricted use. It would appear that in practice the values of warping are not usually known[35] which limits the use of this method as it has been implemented here.

CHAPTER 7

COMPARISON OF METHODS

7.1 Introduction

There are comparisons in the literature, namely Brebbia[23], Brebbia and Dominquez[29], Zienkiewicz, Kelly and Bettes[30], from which the main points in the discussion are taken. Both methods have to be implemented on a computer before use and the quality of the implementation affects performance. The approach here to both methods has been similar - discretization of the boundary or domain was done by program; linear approximation of the boundary with linear interpolation was used in the boundary element method and bilinear shape functions on rectangles were used in the finite element method; integration was done numerically, in general, and advantage was taken of symmetry where convenient.

7.2 Summary of Methods

In Table 7.1 a chart has been made contrasting the main steps in the two methods and commenting on the major differences.

7.3 Comparison of Methods

The main points made by authors when comparing the two methods are summarised and related to the experience gained while doing this work.

Table 7.1 Summary of Methods

Step	FEM / BEM		Comment
1	Basic input		Similar for both
a		steering	Similar for both
b		boundary data	BEM requires the coordinates at which the output is required
c		internal points	
2	Generation of mesh / Discretization of boundary		
	either		
	input		FEM requires nodes covering the whole region and connectivity data. BEM requires nodes on the boundary only - much less data than for FEM
	or		
	automatic mesh generation		Programming is more complicated for FEM owing to connectivity requirements
3	Calculation of stiffness matrix	Integration over elements & assembly of matrix	FEM has a symmetric banded matrix and storage takes account of this. BEM has a full densely packed matrix. It is the most time consuming step in BEM
4	Application of Boundary Data		FEM does Dirichlet problem and particular case of Neumann problem with $\partial u/\partial n = 0$ conveniently, but other boundary conditions involve the evaluation of boundary integrals. BEM deals with all types of boundary conditions most conveniently.

Continued ...

Table 7.1 (continued)

Step	FEM / BEM	Comment
5	Solution of linear equations	Both may use Gaussian elimination but there exist routines to take advantage of the structure in FE matrix.
6	Calculation at internal points	FEM gives solution at nodes throughout region in Step 5. BEM calculates solution at specified points only.
7	Derivatives	FEM uses a difference formula. BEM gives normal derivatives at all boundary nodes.
8	Output of results	FEM outputs solution at all nodes and derivatives as implemented. BEM outputs solution and normal derivatives on the boundary and solutions internally as requested.

Advantages claimed for the boundary element method

(a) The analysis involves only the boundary.

(b) Reduction in core store.

(c) Simpler input data.

(d) Simpler output

(e) When the whole region needs subdivision, the mesh is topologically simpler.

(f) The accuracy of boundary element results is generally greater than that of finite element results.

Comments

(a) Boundary advantage

The reduction of dimension in the solution of the boundary value problem is of particular value to the programmer writing automatic mesh generation routines. These are more necessary in the finite element method where, as the input data is more complex, they help to reduce both the time spent in data preparation and in checking of the element pattern. It is difficult to measure this programming advantage but the lines of code involved are quoted in Table 7.2 as a rough guide.

Table 7.2 Comparison of Subroutine Sizes

Region	Lines of coding	
	BE	FE
Rectangle	33	-
Rectangle with internal mesh	50	62
Circle	8	-
Circle with internal mesh	22	32[33]
Rectilinear figure	25	230

It can be seen that the difference in the size of the mesh generating routines becomes significant when a shape built up out of rectangles is processed.

The finite element routines were then more complex due mainly to their generality. The process described in Section 5.1 involved generating a finite element mesh on the basic rectangles and combining these basic units to form more complex units. The combination was done by storing basic regions on disc files as they were generated and then systematically building up the new region as they were read back into store. Producing a program involves, as well as coding, design and testing both of which were correspondingly more complicated than for the boundary element routines where it was only necessary to subdivide the straight line boundaries of the region. The boundary element routines can cope with any polygonal figure whereas the finite element routines can only generate a mesh over regions that can be subdivided into rectangles. Although this is a particular case, the extra work involved in writing finite element routines is typical of what would be involved in attempting to generate a mesh over a general shape.

(b) Store

As there are fewer nodes in the boundary element method (on a square, 4 n compared with n^2), a reduction in array storage can be expected. The advantage is reduced because the boundary element method has to store the whole densely packed matrix in the solution step whereas the finite element method can take advantage of the banded symmetric structure of its corresponding matrix.

Bettes[31] examines the storage problem and shows that, in a particular range of problems, the finite element method may actually require less store but his breakeven point on the square is at the point where problems enter the practical range. He shows also that the finite element method has an advantage on a rectangle where one side is much longer than the other.

The availability of core store is not usually a problem except when methods are implemented on a small machine. There is a disc based out of core solver developed by Das[34] that can be used with the boundary element method to overcome the problem of the full array in the linear equation solution step.

(c) Input

Without an automatic mesh generator considerably more data is needed to describe a finite element mesh than a boundary element one for the same region. Data in the finite element case has to be given for connectivity as well as for nodes. This method of producing a mesh for a large problem would be expensive in man hours and very error prone.

(d) Output

Output from each method is a matter of implementation. Much more of it can come from a finite element program, especially the mesh description section, but as a lot of it is only used when debugging and testing, it need only appear then. How much output is required when solving a particular problem depends on how the solution is to be used. It would certainly be an advantage in both

methods to arrange to be able to conveniently output either the solution at a few specified points or at points covering the region. The former option is more conveniently arranged in the boundary element method, the latter in the finite element method.

(e) Discretization of the domain

That it is sometimes necessary to subdivide the domain is a disadvantage of the boundary element method. In Section 4.5 it was mentioned that to solve a problem involving Poisson's equation with a general right hand side it would be necessary to subdivide the region to perform the numerical integration and similarly for the calculation of the polar moment of inertia arising in the warping formulation of torsion. What (d) points out is that any convenient mesh is suitable, it is not restricted as the finite element mesh is by nodal connections.

Another approach is to try to find boundary integral representations of the required structural properties. Wood[35] has done this for the torsional stiffness, warping stiffness and shear centre coordinates of a section.

(f) Accuracy

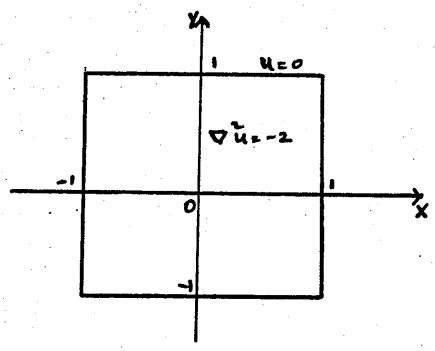


Figure 7.1

Problem 7.1

To solve $\nabla^2 u = 2$ over a square $-1 \leq x \leq 1$ satisfying $u = 0$ on the boundary. Both methods solved the problem over a quarter square to take advantage of the symmetry. The bilinear shape function was used with rectangular elements in the finite element method and linear interpolation in the boundary element method.

Table 7.3 Results obtained by FE and BE Methods

Finite element method

Elts	Mesh Size	Boundary point u(0,0)	u(.5,.5)	u(.75,.75)	$\frac{\partial u}{\partial n}$ (B) (extrapolated) in FEM	Time in etus
16	.25	.5968	.3676	.1501	-1.23	1.49
64	.125	.5912	.3636	.1466	-1.13	2.77
256	.0625	.5894	.3626	.1459	-1.05	9.38
Analytic Solution		.5894	.3622	.1456		
Error %		.0%	.11%	.21%		

Boundary element method

32	.125	.5902	.3625	.1462		2.08
48	.083	.5897	.3624	.1459	-1.1288	4.12
64	.0625	.5895	.3624	.1457	-1.1270	6.97
80	.05	.5895	.3624	.1457	-1.1260	10.97
Error %		.017%	.055%	.07%		

Comment

On this problem it can be seen that the error is dependent on where the point is in the region and on the number of elements. Accuracy of the boundary element method is superior for the same mesh size, particularly

as we get further away from the centres. Accuracy in the finite element method depends on the type and size of elements, how the region is subdivided and the order of shape functions. In the boundary element method it depends on the order of interpolation and the quality of boundary approximation. In two dimensional problems it can also be improved by using closed forms of the integrals.

It is also claimed[23] that, using the boundary element method the derivatives of the original variables may be more accurate than those calculated by the finite element method. This will apply to the normal derivatives on the boundary but it would depend internally on their method of calculation for which a standard method does not appear to exist.

Disadvantages of the Boundary Element Method

(h) The matrices involved in the solution of the linear equations are usually fully populated and not symmetric. This may make the method less computationally efficient.

(i) Restricted to problems where a fundamental solution exists (Section 4.1).

(j) Poisson's equation is more difficult to treat than Laplace's equation (Section 4.5).

Comments

(h) Efficiency

The disadvantage of the structure of the matrix has been discussed. Although it does reduce the efficiency of the method the solution of the linear equations is not usually the major time consumer.

Timings carried out on problem 7.1 with 256 nodes shown in Table 7.3 shows that most time is spent on the element integrations and this is generally accepted by workers on the method. The comparison of timings on the simple problem 7.3 can be seen in Table 7.4. Although the finite element method has four times as many elements the basic solution time is approximately the same as that of the boundary element method. (It needs to be said that the input of the grid by the particular finite element program used here is most inefficient.) The most efficient method to use will depend mainly on what is required from the solution - the boundary element method is clearly at a disadvantage if the solution is required at a lot of internal points.

Table 7.4 Comparison of Timings

	Elts	Nodes	Mesh	Internal Points	Solution etus	Torsional properties etus	Solution at centre
FE	256	289	.0625	-	6.68	9.38	5.894
BE	64	64	.0625	2	6.97		5.895
				4	7.26		
				12		8.42*	

Note* The boundary element method is solving a different boundary value problem here involving Laplace's equation (see Section 6.2).

There have been many comparisons done mainly by boundary element workers on particular problems as they extend the scope of their method. It is fairly normal practice to compare the new solution with one obtained by the finite element method. This, perhaps, highlights the

main benefit of having another solution method. In an area where analytic solutions are rare, confidence in the value of a solution is increased when it compares well with those obtained by other independent methods.

REFERENCES

1. Zienkiewicz, O.C. "The Finite Element Method in Engineering Science". 2nd edition. McGraw Hill, London (1971).

2. Brebbia, C.A. "The Boundary Element Method for Engineers". Pentech Press, London:Plymouth (1980).

3. Schatz, A.H. and Wahlbin, L.B. "Maximum norm estimates in the Finite Element Method on Plane Polygonal Domains, Part 2, Refinements". Maths. Comp. 33, No. 146, pp.465-492.

   Courant, R. and Hilbert, D. "Methods of Mathematical Physics", Vol. 2. Interscience (1962)

4. Op. cit., p.252.

5. Op. cit., p.256.

6. Timoshenko, S.P. and Goodier, J.N. "Theory of Elasticity", 3rd edition. McGraw Hill, Kogakusha, p.311.

   Zienkiewicz, O.C. "The Finite Element Method in Engineering Science". 2nd edition. McGraw Hill, London (1971).

7. Op. cit., p.145.

8. Op. cit., p.42.

9. Op. cit., p.104.

10. Op. cit., p.116.

11. Op. cit., p.149.

12. Op. cit., p.103.

13. Martin, H.C. and Carey, G.F. "Introduction to Finite Element Analysis". McGraw Hill (1973).

14. Baty, J.P. and Pleasants, D.M. "Application of the Collocation and Finite Elements Methods to the Solution of Differential Equations Occurring in Engineering Problems". Innovative Numerical Analysis in Applied Engineering Science, Paris (1977).

15. Rockey, K.C., Evans, H.R., Griffiths, D.W. and Nethercot, D.A. "The Finite Element Method". Granada Publishing (1979).

16. Jaswon, M.A. "Integral Equation Methods in Potential Theory I." Proc. Roy. Soc. A, 275, pp.23-32.

17. Symm, G.T. "Integral Equation Methods in Potential Theory II". Proc. Roy. Soc. A, 275, pp.33-46.

18. Rizzo, F.J. "An Integral Equation Approach to Boundary Value Problems of Classical Elastostatics". Quart. Appl. Math. 25.

19. Cruse, T.A. and Rizzo, F.J. "A Direct Formulation and Numerical Solution of the General Transient Elastodynamic Problem 1. J. Math. anal. Appln. 22, pp.244-259.

Cruse, T.A. "A Direct Formulation of the General Transient Elastodynamic Problem II". J. Math. Anal. Appln. 22, pp.341-355.

20. Cruse, T.A. "Application of the Boundary-Integral Equation Method in Solid Mechanics". Variational Methods in Engineering, Vol. II, Proc. Int. Conf. Variational Meth. Engng. Southampton (1972).

Brebbia, C.A. "The Boundary Element Method for Engineers", Pentech, London.

21. Op. cit., p.52.

22. Op. cit., p.55.

23. Op. cit., p.183.

24. Danson, D.J. "BEASY a Boundary Element Analysis System." Proc. 4th Int. Seminar, Southampton (1982).

25. Courant, R. and Hilbert, D. "Methods of Mathematical Physics". Vol. 2, Interscience, 1962, p.244.

26. Brebbia, C.A. and Walker, S. "Introduction to Boundary Element Methods. Recent Advances in Boundary Element Methods". ed. C. A. Brebbia, Pentech (1978).

28. Jaswon, M.A. and Symm, G.T. "Integral Equation Methods in Potential Theory and Elastostatics". Academic Press (1977).

29. Brebbia, C.A. and Dominguez, J. "Boundary Element Methods versus Finite Elements". "Applied Numerical Modelling". ed. Brebbia, C.A., Pentech (1977).

30. Zienkiewicz, O.C., Kelly, D.W. and Bettess, P. "The Coupling of the Finite Element Method and the Boundary Solution Procedures". Int. J. Num. Methods Engng. 11, pp.355-375 (1977).

31. Bettess, P. "Operation Counts for Boundary Integral and Finite Element Methods". App. Num. Modelling (1981), pp.306-308.

32. Mukherjee, S. and Morjaria, M. "The Comparison of Boundary Element and Finite Element Methods in the Inelastic Torsion of Prismatic Shafts". App. Num. Modelling (1981), pp.1576-1587.

33. Fenner, R.T. "Finite Element Methods for Engineers". Macmillan (1975).

34. Das, P.C. "A Disc Based Block Elimination Technique used for the Solution of Non-symmetrical fully Populated Matrix Systems encountered in the Boundary Element Method". Recent Advances in Boundary Element Methods. C. A. Brebbia (ed.), Pentech (1978).

35. Wood, D.J. "Determination of the Torsional Properties of a Plane Section using Boundary Integral Techniques." App. Math. Modelling (1980), Vol. 4.

36. Strang, G. and Fix, G.J. "An Analysis of the Finite Element Method". Prentice Hall (1973), p.263.

37. Symm, G.T. "Treatment of Singularities in the Solution of Laplace's Equation by an Integral Equation Method". NPL Report NAC 31 (1973).

38. Whiteman, J.R. and Papamichael, N. "Numerical Solution of Two dimensional Harmonic Boundary Problems Containing Singularities by Conformal Transformation Methods". Brunel University, Dept. of Maths Report No. TR/2 (1971).

39. Motz, H. "The Treatment of Singularities of Partial Differential Equations by Relaxation Methods". Quart. Appl. Maths, (1946), 4, pp.371-377.

40. Saint Venant "Memoires Savants Entrangers, I." (1855)

41. Timoshenko, S.P. and Goodier, J.N. "Theory of Elasticity". 3rd edition, McGraw-Hill, Kogakushe.

42. Op. cit., p.126.

43. Op. cit., p.236.

44. Op. cit., p.237.

45. Op. cit., p.293.

46. Op. cit., p.298.

47. Jaswon, M.A. and Ponter, A.R. "An Integral Equation Solution of the Torsion Problem". Proc. Roy. Soc. Ser. A, 273, 237 (1963).

48. Valliappan, S. and Pulmann, V.A. "Torsion of Nonhomogeneous and Anisotropic Bars". J. Struct. Div., Proc. A.S.C.E., Vol. 100, No. ST1 January 1974, pp.288-295.

APPENDIX 1 FINITE ELEMENT PROGRAMMING

A1.1 Introduction

Program Function

This set of routines is used to solve the two dimensional boundary value problem governed by Poisson's partial differential equation over a region R, where the solution satisfies a boundary condition of the form

$$u = c \text{ or } \frac{\partial u}{\partial n} = 0$$

on the boundary, B, of R using the finite element method. Automatic element generation is provided for polygonal regions that are built up from rectangles.

Program Names

FEMREG subdivides a rectangle into rectangular or triangular elements.

FEMCBN combines regions with their element meshes to give the mesh for a larger region.

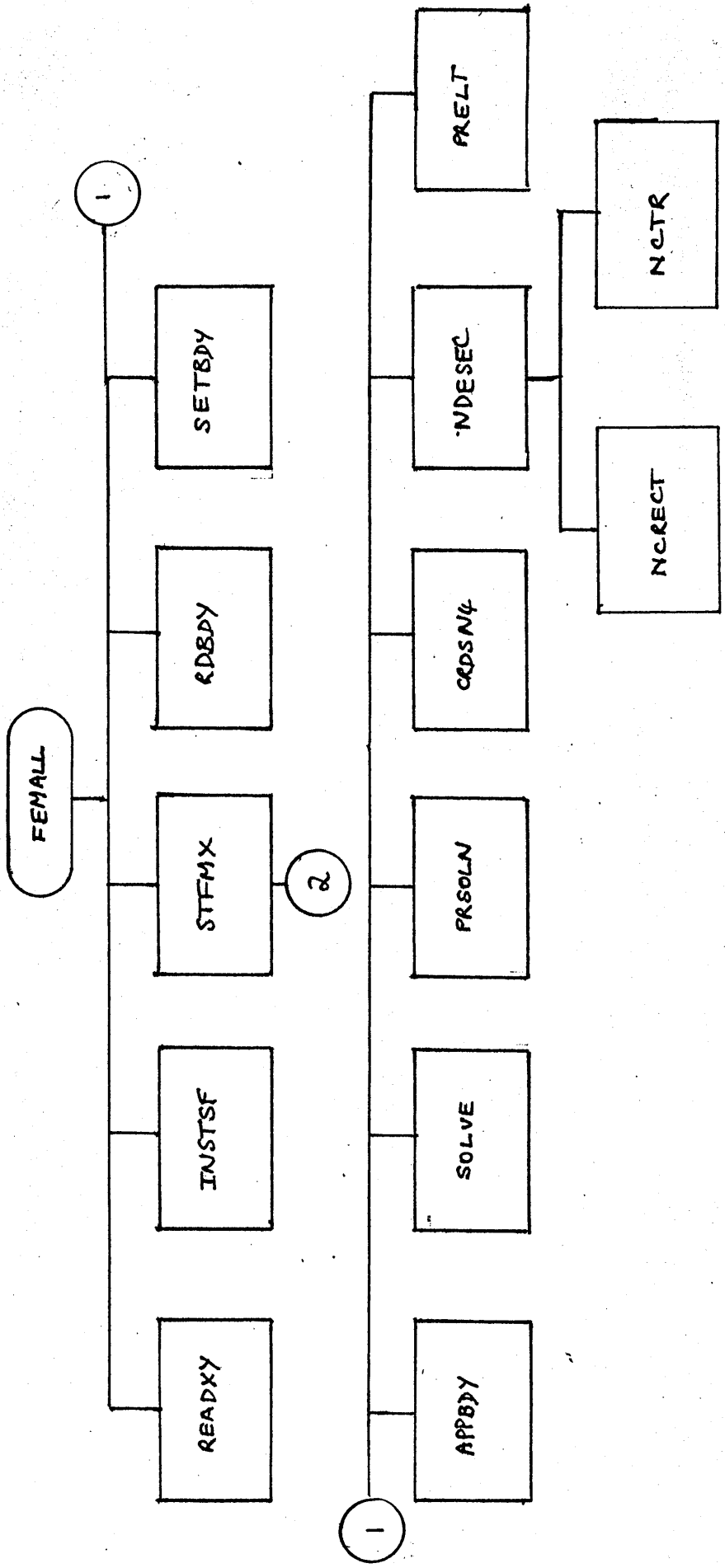
FEMALL solves the boundary value problem by the finite element method

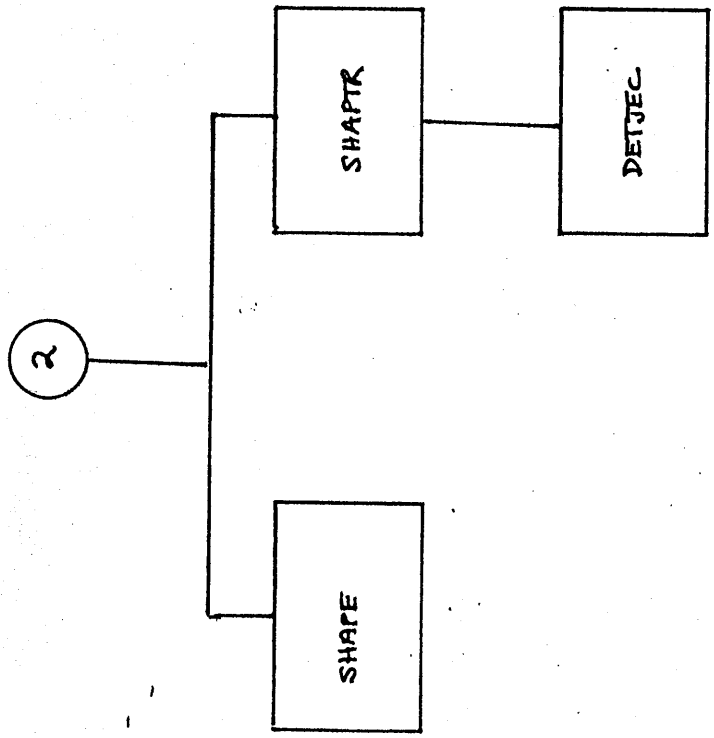
Language and Operating System

FORTRAN under TOPS on a DEC-20.

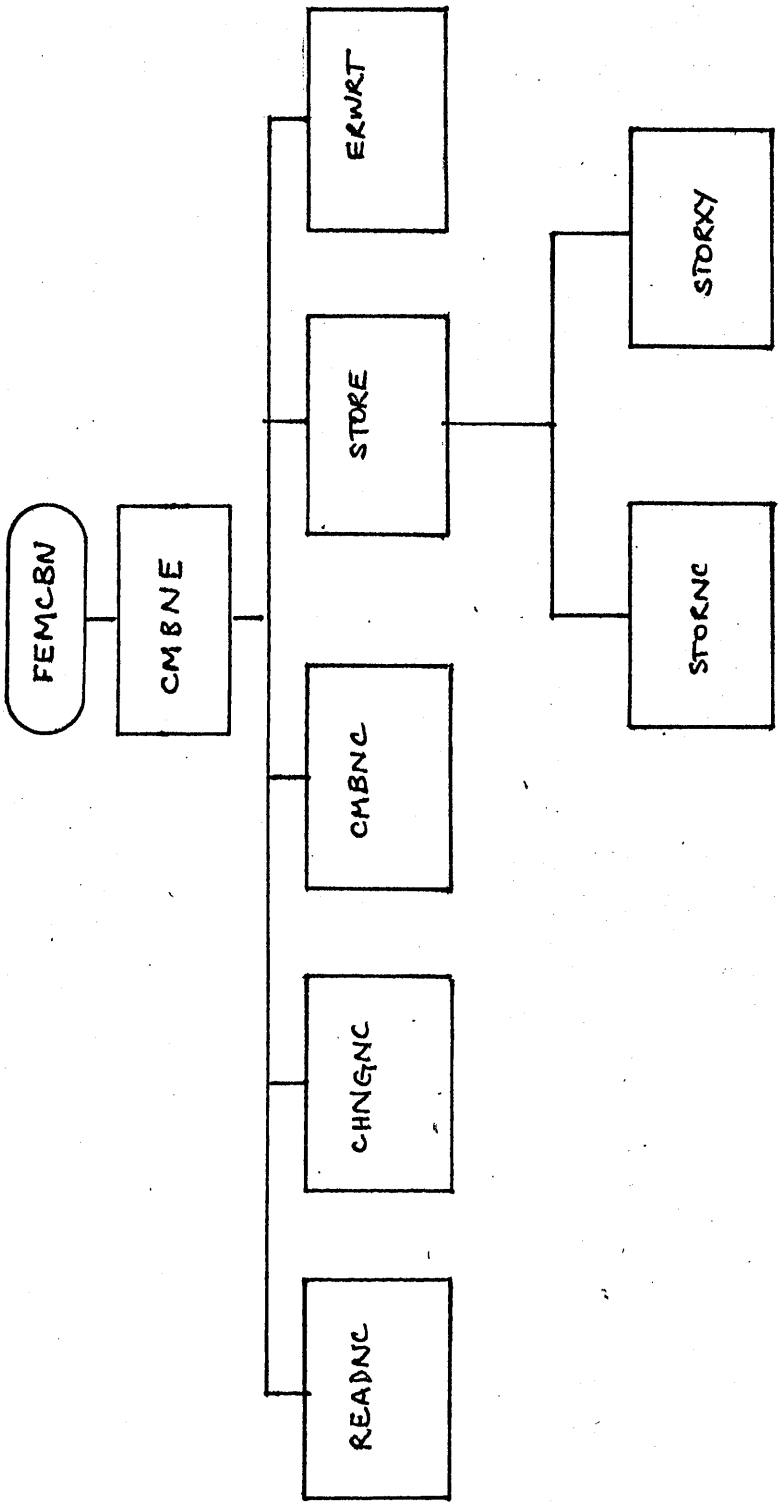
A1.2 Structure of Routines

A1.21 Structure FEMALL

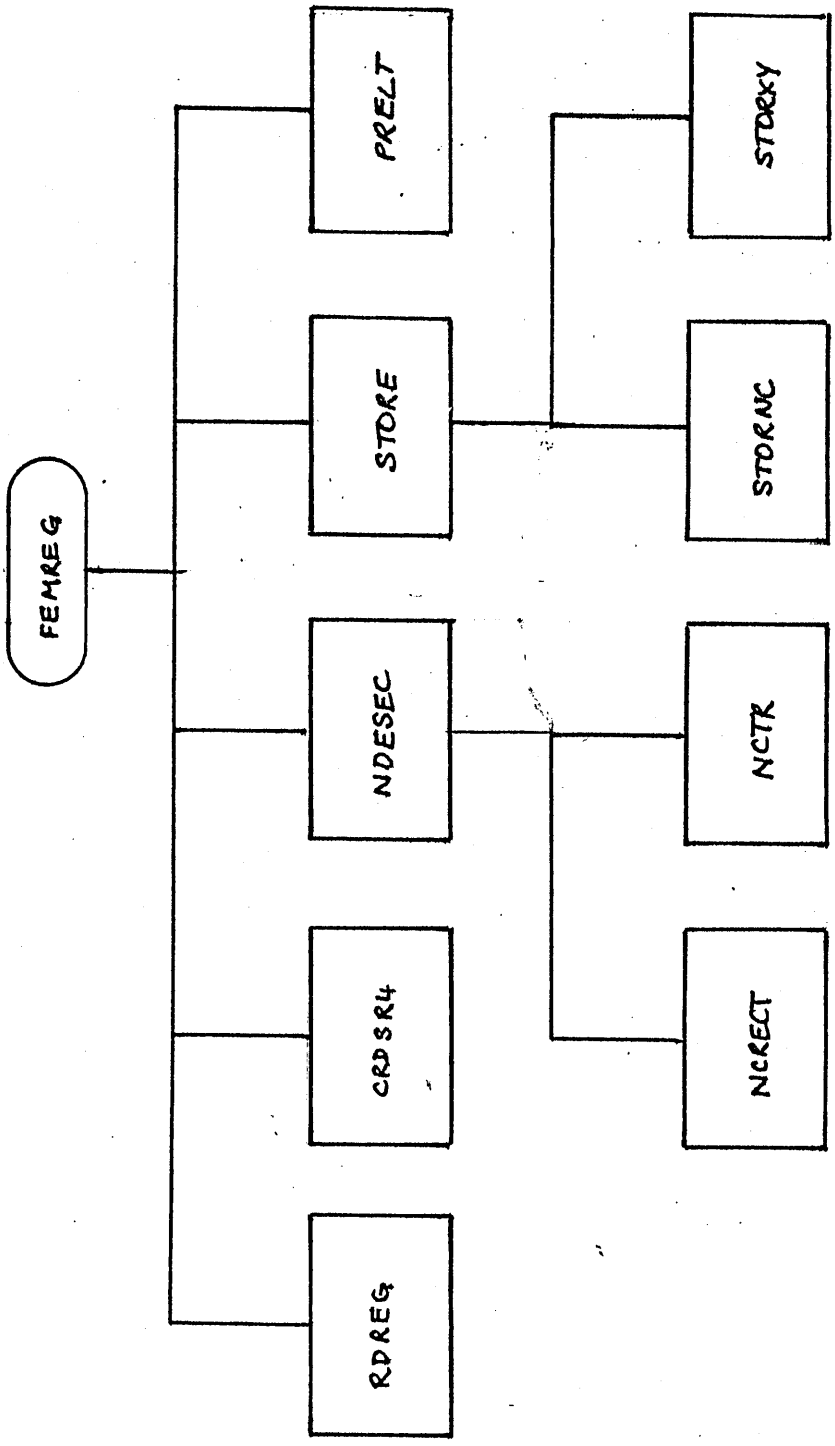



2
SHAPTR
DETJEC
SHAPE

A1.22 Structure FEMCBN



A1.22 Structure FEMREG



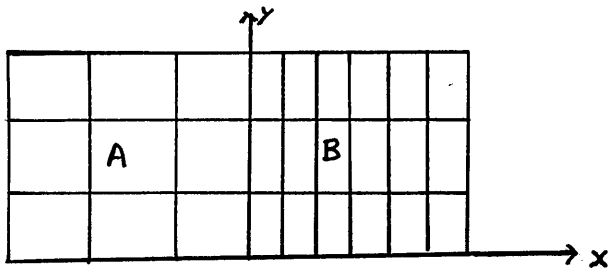
A1.3 Main Programs

FEMALL

This is the main finite element program which calls other routines to perform the main steps described in Section 3.6. It reads the data selecting the element type, shape function and the level of numerical integration. Subroutines are called - READXY to read mesh data, STFMX to assemble the stiffness matrix, APPBDY to apply the boundary condition, SOLVE to solve the linear equations and PRSOLN to print the solution. If the region involved is a rectangle it is then possible to generate a finer mesh and find the solution using it.

FEMCBN

This is the main program in the set of routines that are used to combine two regions. It reads data specifying the regions and their files and then opens the two input files containing regional data and an output file to which the resulting mesh is written.



The combining routines take the nodal description of two regions with a common boundary line to give the data for the combined region. The regions that can be processed must have been built up from rectangles which were originally divided into a rectangular element mesh with one direction of the subdivision parallel to the y-axis and the increment in y at the points on the interface the same in both regions. Some examples are shown in Figure A1.1.

Along the join the nodes have to match except where there is a permissible displacement occurring when one region is moved by an integral number of units up or down as in Figure A1.1a and b.

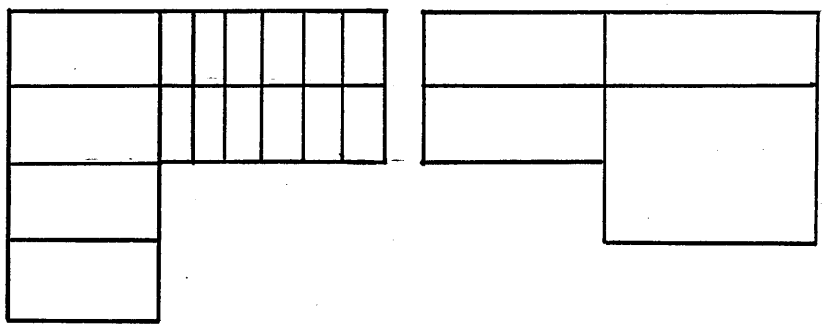


Figure A1.1

FEMREG

This is the main program for subdividing a rectangle into elements. It opens input and output files and calls other subroutines to read data, generate the mesh and store the results on disc file.

A1.4 Subroutines

APPBDY

This subroutine uses the boundary values stored in the array,BC,to apply the boundary condition by the method described in Section 3.6. When a node is on the boundary the coefficients not on the diagonal affecting the node are set to zero, the diagonal value set to 1 and the values in the solution array, B, modified by the boundary value for the positions involved.

CHNGNC

This subroutine changes the numbers of the nodes in the relevant nodal connection array after two rows have been joined. The nodal numbers in the right hand region are increased by the specified amount. As the nodal numbers relating to original regions are stored as negative numbers those relating to the new region are stored as positive numbers to distinguish between the two sets.

CMBNC

This subroutine combines the two nodal connection arrays, NC1 and NC2, to form the array, NC, relating to the new region. The complete arrays NC1 and then NC2 are copied into the array NC changing the sign on any negative nodal number.

CMBNE

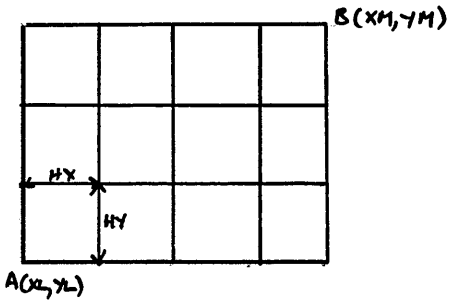
This subroutine checks that the input files contain data about the regions specified for combination and reads the headers into the arrays, MSREG, and nodal connection data into the arrays, NC1 and NC2.

The coordinates of all nodes with the same y-coordinate are read from two input files into the arrays X and Y, first from one file and then from the other. In cases where there is a displacement of one region with respect to the other, as in Figure A1.1, the nodal coordinates of the lower region are read row by row until a value of the y coordinate is reached involving a join after which the process is continued as above. When all rows on the join have been processed, if one region has not been read completely then it is read row by row until its end.

Whenever two rows have been joined the numbers of the nodes are altered by routine, CHNGNC, in the nodal connection arrays to their values in the new region.

Finally the header data for the combined region is assembled in the array MSREG. The nodal connection data is assembled to relate to element order in the resulting region by the routine, CMBNC and the data about the mesh is output to disc by the routine, STORE.

CRDSR4



This subroutine divides the rectangular regions into rectangular elements storing the corner coordinates in arrays X and Y. The number of divisions along each side of the rectangle is used to find the element size, HX x HY. The coordinates of the nodes are calculated, in order, starting from the corner A(XL,YL) moving from left to right at the same horizontal level stepping a length HX until x-coordinate equals XL after which the y coordinate is increased by an amount YL. The coordinates of nodes at this level are calculated, again moving from left to right and the process is repeated until the upper right hand corner (XM, YM) is reached.

The number of elements, NE, and the number of nodes, NN are stored in common.

ERWRT

This subroutine writes an error message when an error is detected by the combination routine, CMBNE. The program stops after such a message.

INTSTF

This subroutine sets the array, JP, that holds the first significant item in a column of the stiffness matrix and the array, NDA containing the position in which each diagonal element in the stiffness matrix has been stored in the array, A. The arrays A and B in which the linear equations are assembled and stored are set to zero.

NCRECT

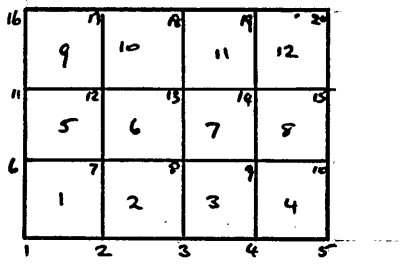


Figure A1.2

This subroutine stores for each element the nodal numbers of its corner nodes in the array NC. In each element the nodal number is calculated in the lower left hand corner of the element, e, that is

$$N_{ij}(e) = (j-1) \times (N+1) + i$$

and the other nodal numbers in the element e follow by using the fact that there are N+1 nodes in a row.

NCTR

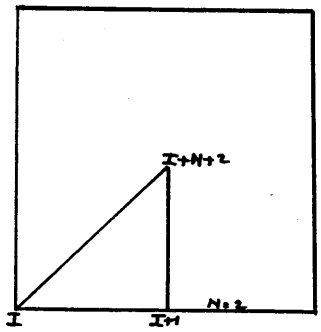


Figure A1.3

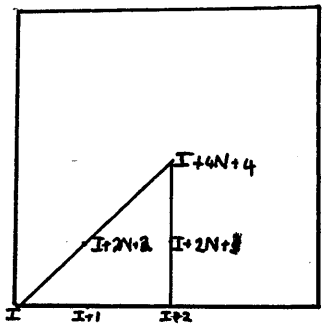


Figure A1.3b

This subroutine works out the numbers of the nodes in each element for both 3 node and 6 node triangles and stores them in element order in the array NC.

For the 3 node triangle the original rectangle is subdivided into an N x M rectangular mesh. Each pair of triangular elements is processed in turn recording the lower left hand corner position first and then finding the other vertices as shown in Figure A1.3.

For the 6-node triangle, the original rectangle is divided into a 2N x 2M rectangular mesh. Again each pair of elements is taken finding the nodal position of the vertices and midpoint nodes from the left hand corner position as shown in Figure A1.3.

The diagonals used to create the triangular mesh are parallel and the requested direction is specified in the original input data.

NDESEC

This subroutine calls others to find the nodal connections according to element and shape function type.

PRELT

This subroutine prints headings and details of the boundary value problem being solved together with those of the elements and the shape functions used.

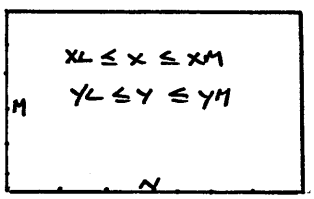
PRSOLN

This subroutine prints the nodal values from the array, B, and the partial derivatives of the solution found at the centroid of each element using a difference formula. It calculates the torsional rigidity using the solution values in B and the element mesh to do the numerical integration.

RDBDY

This subroutine reads the boundary data. It may be specified by the value the solution takes at each nodal point or by the value it takes at all nodes on a specified straight line boundary.

RDREG



This subroutine inputs data defining the rectangle R to be subdivided and specifying the number of elements along each side of R, N x M, as well as program steering data. The data is stored in common blocks.

READNC

This subroutine finds the number of nodes associated with the element selected and the number of coordinates for each point when expressed in local coordinates from block data. For complex regions where more than one element type may occur, the maximum value of these variables over all element types involved is used. The nodal connection map is read into the array NC in blocks of 256 words. The nodal numbers are made negative to indicate they relate to an input region.

READXY

This subroutine reads a data file containing the mesh details. The headers are read and stored in common then the subroutine, READNC, reads the nodal connection data. The nodal coordinates are read and stored in the arrays X,Y.

SETBDY

This subroutine checks each node to see whether it is on the boundary and, if it is, stores the value the solution is to take in the array, BC and, if not, an indicator is set in array BC.

SHAPE

This subroutine calculates and stores in the array, SH, the values of the bilinear shape functions and their derivatives at the specified point in a rectangular element. The functions are expressed in nodal coordinates.

SHPTR

This subroutine calculates and stores in the array, SH, the values of either the 3 node linear or 6 node quadratic shape functions and their derivatives used with triangular elements. Area coordinates are used to specify the functions while the area of the triangular element is found using the subroutine DETJAC.

SOLVE

This subroutine solves a set of linear equations using Gaussian elimination. The coefficients of the equations have to be unpacked from the array A using the pointers in arrays JP and NDA. The right hand side of the equations are stored on entry in the array B and on exit B contains the solution.

STFMX

This subroutine calculates the element stiffness matrix in the array, ES and the contribution to the load vector in the array, BS. The process is described in Section 3.6 and is dependent on the element being used. At each numerical integration point the values of the shape functions are needed and are placed in the array SH by either the subroutine SHAPE or SHAPTR. After each element contribution has been calculated it is added to the main stiffness array, A, positioning it according to the nodes in the element found from the nodal connection array, NC.

STORE

This subroutine writes the header information to the data file, INFO, and calls other routines to store geometric and

connectivity data for the region, NOREG. The header data is written from common data for a basic region (denoted by a positive region number and generated from input data) and from the array, MSREG, for a complex region (denoted by a negative region number and generated by combining other regions).

STORNC

This subroutine writes the information about element types and nodal connections within the region to the output file. If more than one element type is used then a list of element types is first written from array, LELT. The contents of the array, NC, are stored in blocks of 256 words.

STORXY

This subroutine writes the limits of the region and the coordinates of the nodes, in order, from the arrays X and Y. This is done by writing a variable length record of all points with the same Y coordinate preceded by a one word record containing the length of this record.

APPENDIX 2 BOUNDARY ELEMENT PROGRAMMING

A2.1 Introduction

Program Function

This set of routines is used to solve the two dimensional boundary value problem governed by Poisson's partial differential equation over a region, R where the solution satisfies a boundary condition of the form,

$$a\ u + b\ \frac{\partial u}{\partial n} = 0,$$

on the boundary, B, of R using the boundary element method. The boundary discretization can be done fully by the program for polygons and circles.

Program Name

BMAIN

Language and Operating System

FORTRAN under TOPS on a DEC-20.

Acknowledgement

This set of routines is based on those published by Brebbia[2].

A2.2 Main Program

BMAIN

This program opens the data file and calls other routines - BINPUT to read data, BFMAT to organise equations, BSLNPD to solve

the linear equations, BINTER to calculate the solution at requested internal points, and BOUTPT to output. The program solves Laplace equations but the routine, BPOISS is used to add the particular solution when solving Poisson's equation. When solving the torsion problem the extra routines, BWARP and BTRIGD, are used to set the boundary values and calculate rigidity respectively.

A2.3 Subroutines

BCXCY

This subroutine generates a rectangular mesh over a rectangle that corresponds on the boundary to the boundary discretization. If the rectangle is centred at the origin it is possible to have the mesh over the upper right hand quarter only.

BDERIV

This subroutine amends the normal derivatives when Poisson's equation is being solved over rectangles, circles or polygons.

BFMAT

This subroutine uses routines published in Brebbia[2]. It assembles the linear equations in the arrays G and H calling INTE and INLO to do the integrations along the boundary elements. The matrices are rearranged so all the coefficients referring to unknown boundary values of u and $\frac{\partial u}{\partial n}$ are in matrix G and those referring to known values in H. The right hand side of the linear equations is calculated using H and the boundary values that were input and in array FI. The details of this process are described in Section 4.6.

If Poisson's equation is being solved the subroutine starts by modifying the boundary values specified to allow for the particular integral.

BINLO

This subroutine is taken from Brebbia[2]. It does the integration for the diagonal terms of G and H using an exact integration as described in Section 4.6.

BINPUT

This subroutine inputs program control data and calls the routines BRRECT, BRDCIR and BRSLIN to do boundary discretization, if requested. Otherwise it inputs the coordinates of the boundary nodes to arrays X and Y. It reads the coordinates of the internal points at which the solution is required into arrays CX and CY. The boundary values at all nodes are input or if a constant value is taken on the boundary, the values at all nodes may be set by program.

BINTE

This subroutine is taken from Brebbia[2]. It does the integrations for the off diagonal coefficients in the matrices G and H using numerical quadrature as described in Section 4.6.

BINTER

This subroutine uses routines published in Brebbia[2]. It puts the solution values in array, F1 and the normal derivatives in array DF1 and calculates the solution at the specified internal points in array,SOL.

BOUTPT

This subroutine is based on routines published in Brebbia[2]. It outputs the results of the problem preceded by the details of the discretization of the boundary and the boundary values.

BRDCIR

This subroutine approximates a circular boundary by a set of equal length cords. The nodal points are calculated by finding the angle that subtends each cord using the number of elements specified.

BRRECT

This subroutine subdivides the boundary of a rectangle into elements. The number along each side is specified allowing the size of equal length elements to be calculated. The nodal coordinates are stored in arrays XR, YR. If requested the subroutine BCXCY is called to generate a mesh of internal points.

BRSLIN

This subroutine subdivides a polygonal boundary, each boundary of which has been specified together with the number of elements required on it.

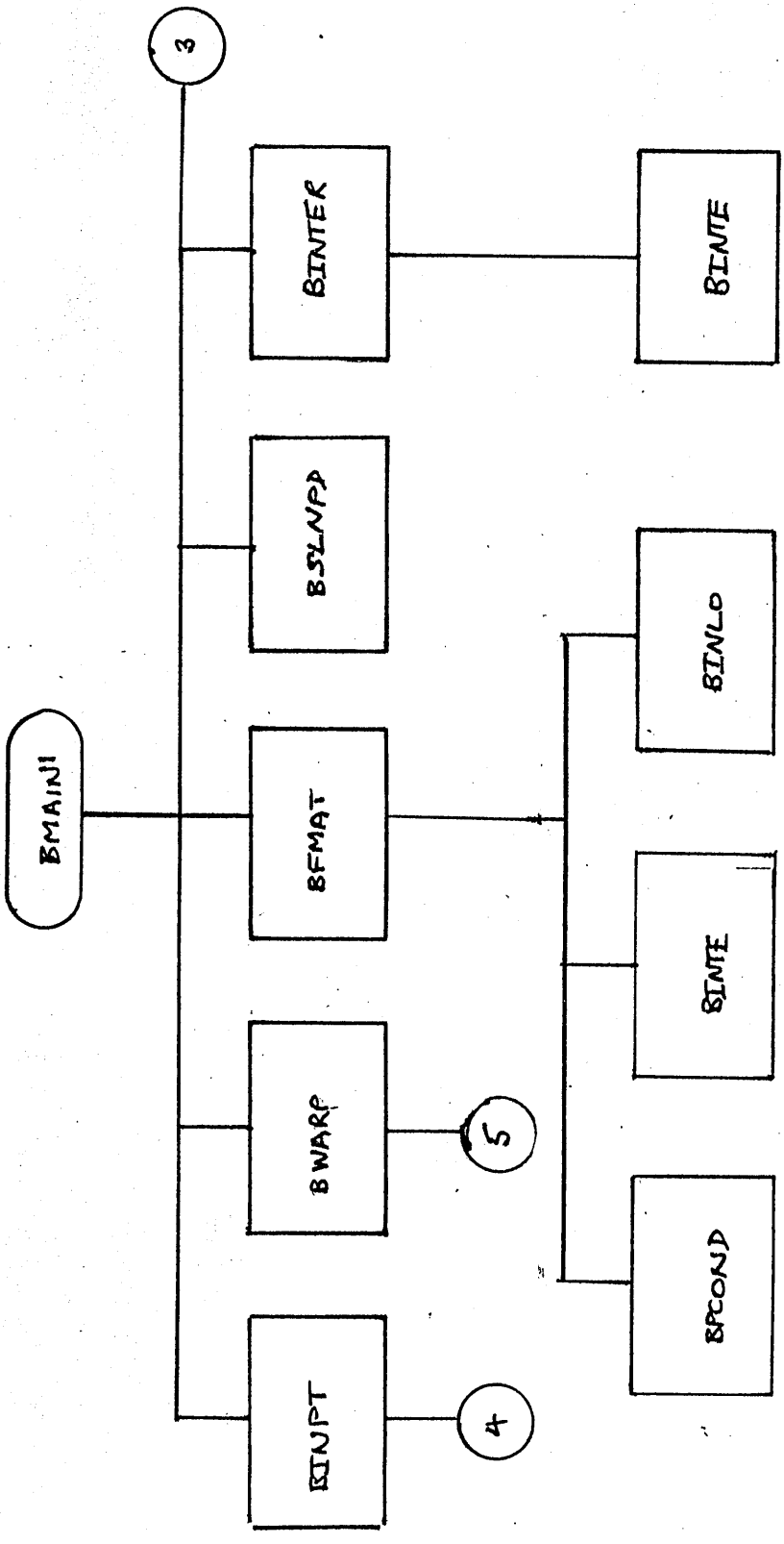
BSLNPD

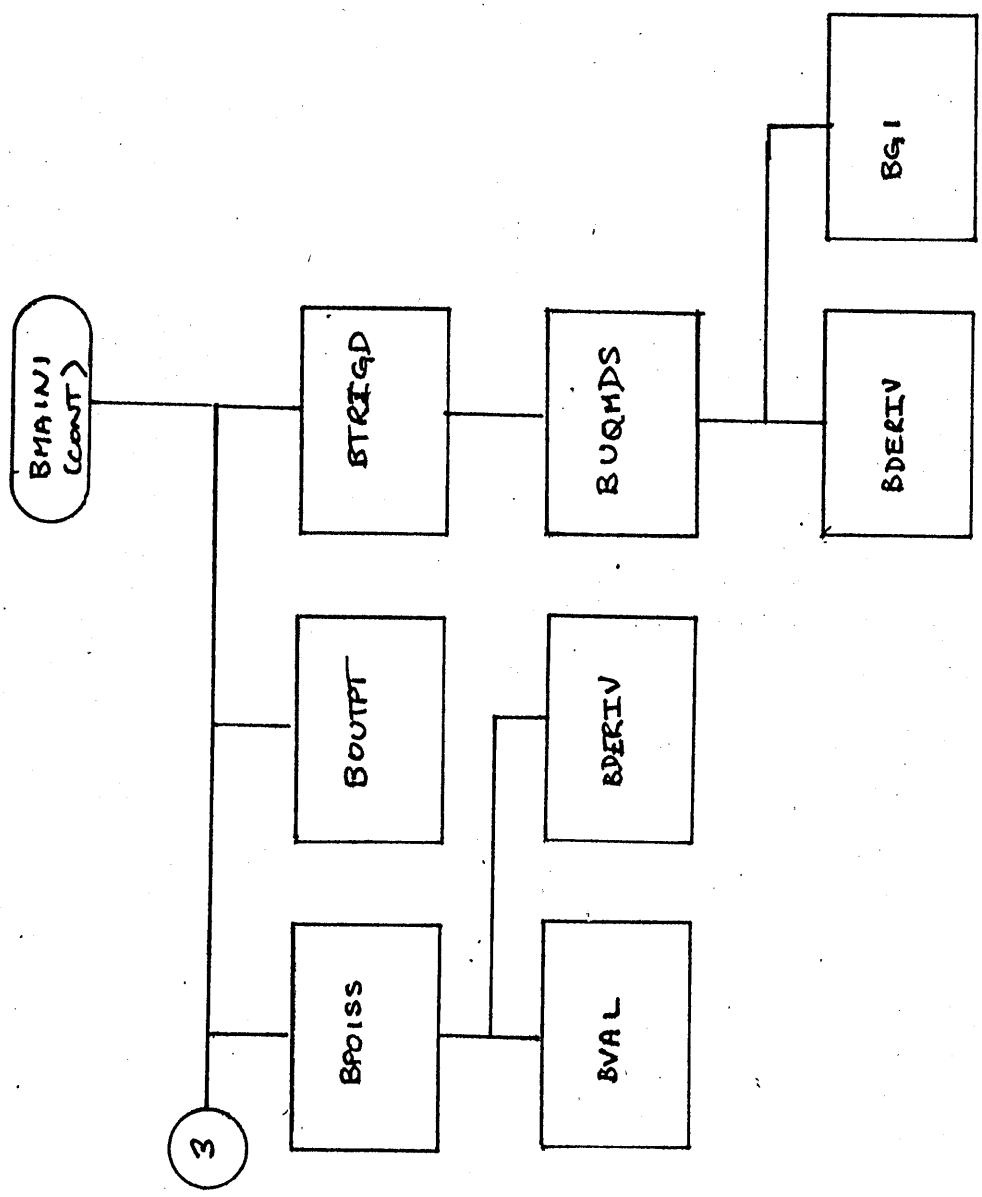
This subroutine is taken from Brebbia[2]. It solves the linear equations whose coefficients are stored in array A and right hand side in B using Gaussian elimination.

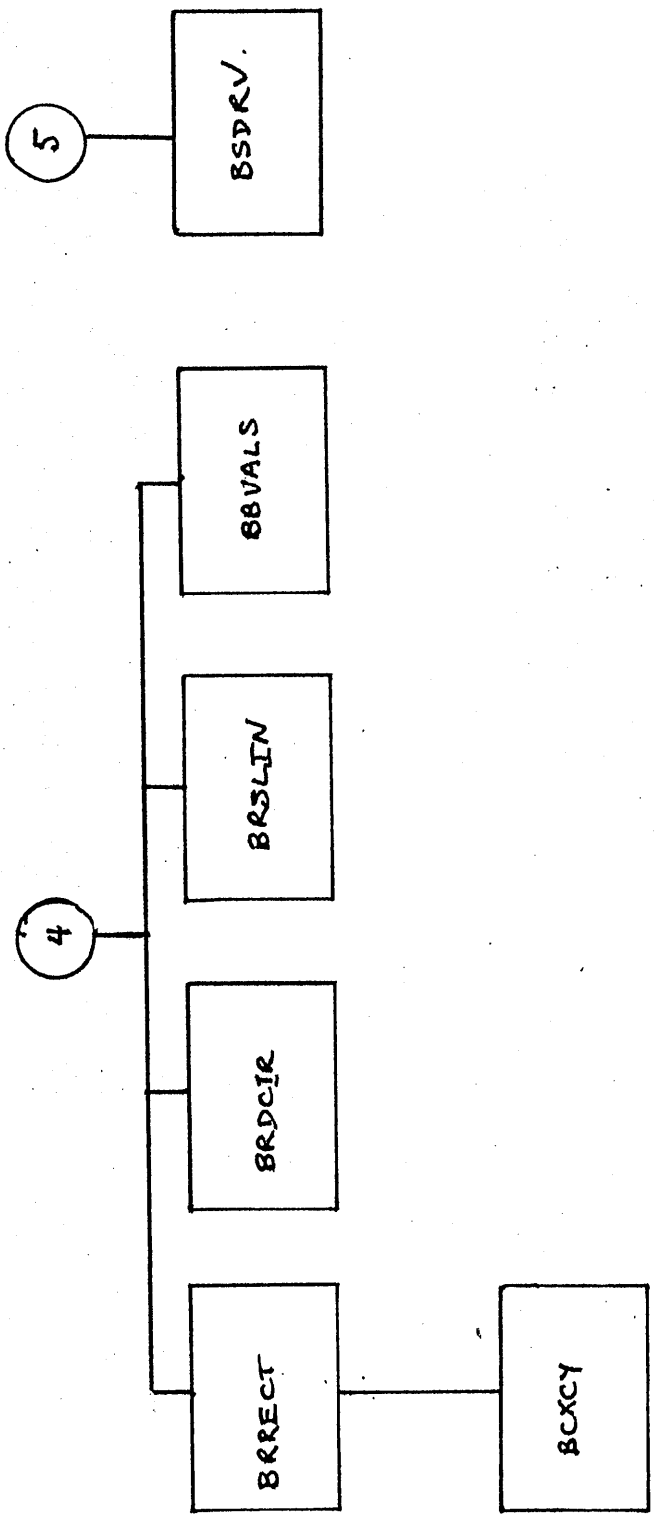
BVAL

This function calculates the value of the particular solution added to Poisson's equation.

A2.4 Structure of BMAIN1




BMAIN1 (CONT)
BTRIGD
BUQMDS
BQ1
BDERIV
BOUTPT
BDERIV
BDISS
BVAL
3


5
BSDRV.
4
BBVALS
BRSLIN
BRDCIR
BRRECT
BCXCY